P. GUEULLE

ROBOTISEZ VOTRE ZX 81



MICRO SYSTEMES

ETSF

Collection
MICRO-SYSTEMES

ROBOTISEZ
VOTRE
ZX 81

Photo de couverture :
Un sciage très soigneux permet de séparer en deux parties la coquille supérieure du boîtier du ZX 81 : un grand pas vers des interventions sérieuses sur le circuit interne de l'ordinateur !

« La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que « les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'Art. 40). Cette représentation ou reproduction, par Art. 425 et suivants du Code pénal ».

© 1984 — E.T.S.F.

ISBN 2-85535-076-X

ISSN 0759-9498

Patrick Gueulle
ingénieur EFREI

ROBOTISEZ VOTRE ZX 81

Diffusion
EDITIONS TECHNIQUES ET SCIENTIFIQUES FRANCAISES
2 à 12, rue de Bellevue, 75940 Paris Cedex 19

COLLECTION ETSF MICRO-SYSTEMES

1 - A. VILLARD et M. MIAUX, *Un microprocesseur pas à pas*
2 - A. VILLARD et M. MIAUX, *Systèmes à microprocesseur*
3 - P. GUEULLE, *Maîtrisez votre ZX 81*
4 - E. FLOEGEL, *Du Basic au Pascal*
5 - P. COURBIER, *Vous avez dit Basic ?*
6 - M. MARCHAND, *Vous avez dit micro ?*
7 - P. GUEULLE, *Pilotez votre ZX 81*
8 - M. JACQUELIN, *La micro-informatique et son A-B-C*
9 - M. OURY, *Maîtrisez les TO 7 et TO 7-70*
10 - P. GUEULLE, *Pilotez votre Oric 1 + Atmos*
11 - P. JOUVELOT, D. LE CONTE DES FLORIS,
Système d'exploitation et logiciel de base des micro-ordinateurs

COLLECTION POCHE-INFORMATIQUE

1 - G. ISABEL, *50 programmes pour ZX 81*
2 - P. GUEULLE, *Montages périphériques pour ZX 81*
3 - C. GALAIS, *Passeport pour Applesoft*
4 - R. BUSCH, *Passeport pour Basic*
5 - M. ROUSSELET, *Mathématiques sur ZX 81*
6 - C. GALAIS, *Passeport pour ZX 81*
7 - G. PROBST, *50 programmes pour CASIO FX-702 P et FX-801 P*
8 - G. PROBST, *60 programmes pour CASIO PB-100*
9 - M. SAAL, *Utilitaires pour ZX 81*
10 - C. GALAIS, *Passeport pour Commodore 64*

SOMMAIRE

Chapitre V
PROGRAMMONS NOS MEMOIRES MORTES

Chapitre VI
UN AFFICHEUR AUTONOME POUR LE ZX 81

Chapitre VII
UNE CARTE SONORE ADAPTABLE AU ZX 81

Chapitre VIII
DES APPLICATIONS PRATIQUES

Chapitre IX
UNE CARTE MICROPROCESSEUR COMPATIBLE ZX 81

Avant-Propos

A l'heure où nous écrivons ces lignes, le nombre de ZX 81 en circulation dans le monde doit approcher du million, alors que des séries confortables sortent toujours des usines Sinclair et Timex.

Le ZX 81 est une merveilleuse machine d'initiation, son succès fulgurant l'a amplement démontré.

Cependant, les vrais passionnés de l'informatique (et ils sont légion parmi les victimes du ZX 81 !) se laissent très souvent séduire par des machines plus sophistiquées, dont le marché commence à être inondé.

Vendre d'occasion son fidèle compagnon des premiers pas en informatique est généralement déchirant et, qui plus est, une très mauvaise affaire aux cours actuels.

Beaucoup de ZX 81 font ainsi valoir leurs droits à la retraite au fond d'un placard poussiéreux, et c'est fort dommage !

Même si un usage intensif les a quelque peu défraîchis, ces petits ordinateurs représentent toujours une sérieuse puissance de travail, qui peut facilement être mise à contribution pour introduire au foyer une certaine forme de « robotique ».

Seulement, on hésite à mobiliser un récepteur TV et un magnétophone pour automatiser un aquarium ou un système d'alarme. On se méfie également des coupures de courant, qui risquent d'effacer un programme pourtant vital...

Histoire ancienne que tout cela !

Quelques accessoires faciles à construire par soi-même à peu de frais permettent de transformer le ZX 81 en un véritable robot ou automate à microprocesseur qui se suffira entièrement à lui-même aussi longtemps que nécessaire. Toutes les informations théoriques et pratiques nécessaires sont réunies dans ce petit livre qui, nous le souhaitons, sauvera de l'oubli bien des ZX 81 encore bien jeunes pour cesser tout service actif !

Chapitre 1

Autopsie du ZX 81



La mise à l'étude d'extensions matérielles pour le ZX 81 exige une connaissance sinon totale, du moins satisfaisante de l'organisation interne de la machine.

Une auréole de mystère a toujours entouré les produits Sinclair, sans doute dans le but de protéger autant que possible des techniques de pointe hautement vulnérables. C'est compter sans la persévérance des passionnés !

De patientes recherches, d'innombrables recoupements entre des expériences et des lectures de publications britanniques permettent maintenant aux membres de ce que certains ont baptisé le « clan Sinclair », de se faire une idée beaucoup plus nette de ce qui se passe sous le petit capot noir...

Organisation générale d'un système à microprocesseur

Le ZX 81 n'est qu'un cas particulier de système à microprocesseur, mais quel cas !

Une organisation commune se retrouve dans tous les systèmes microprogrammés, seules des variantes à vrai dire mineures permettant de les distinguer (*fig. 1-1*).

Si l'on excepte les organes « secondaires » que sont, par exemple, les circuits d'alimentation et d'horloge, on peut scinder n'importe quel système en trois grandes parties :

• **l'unité centrale,** c'est-à-dire le microprocesseur lui-même, chargée d'exécuter tous les traitements d'information proprement dits. Un jeu d'instructions, propre à chaque modèle de microprocesseur, fixe de façon limitative la panoplie d'opérations élémentaires disponibles. C'est cela, le *langage machine* que l'on utilisera...

Outre les circuits logiques de traitement, dont le comportement est dicté par les instructions exécutées, le microprocesseur contient un certain nombre de *registres* capables de stocker temporairement des informations.

La plupart des instructions opèrent sur des données présentes dans des registres, et rangent le résultat également dans un registre.

Le microprocesseur Z 80 A, autour duquel est bâti le ZX 81, est l'une des unités centrales possédant le plus de registres, et le plus large jeu d'instructions : un sérieux gage de souplesse d'emploi !

• **la mémoire** se décompose elle-même en deux parties : la *mémoire morte* ou ROM, qui contient le *programme* machine à exécuter ainsi que des données figées (par exemple la forme des caractères utilisés par le ZX 81, ou encore la valeur de pi). Cette mémoire est *programmée* une fois pour toutes en usine, et son contenu ne peut en aucun cas être altéré par l'utilisateur.

La *mémoire vive* ou RAM sert de « bloc-notes » au microprocesseur, qui y rangera à son gré (ou plutôt au gré du programme !), les informations que les registres internes ne suffiraient pas à abriter.

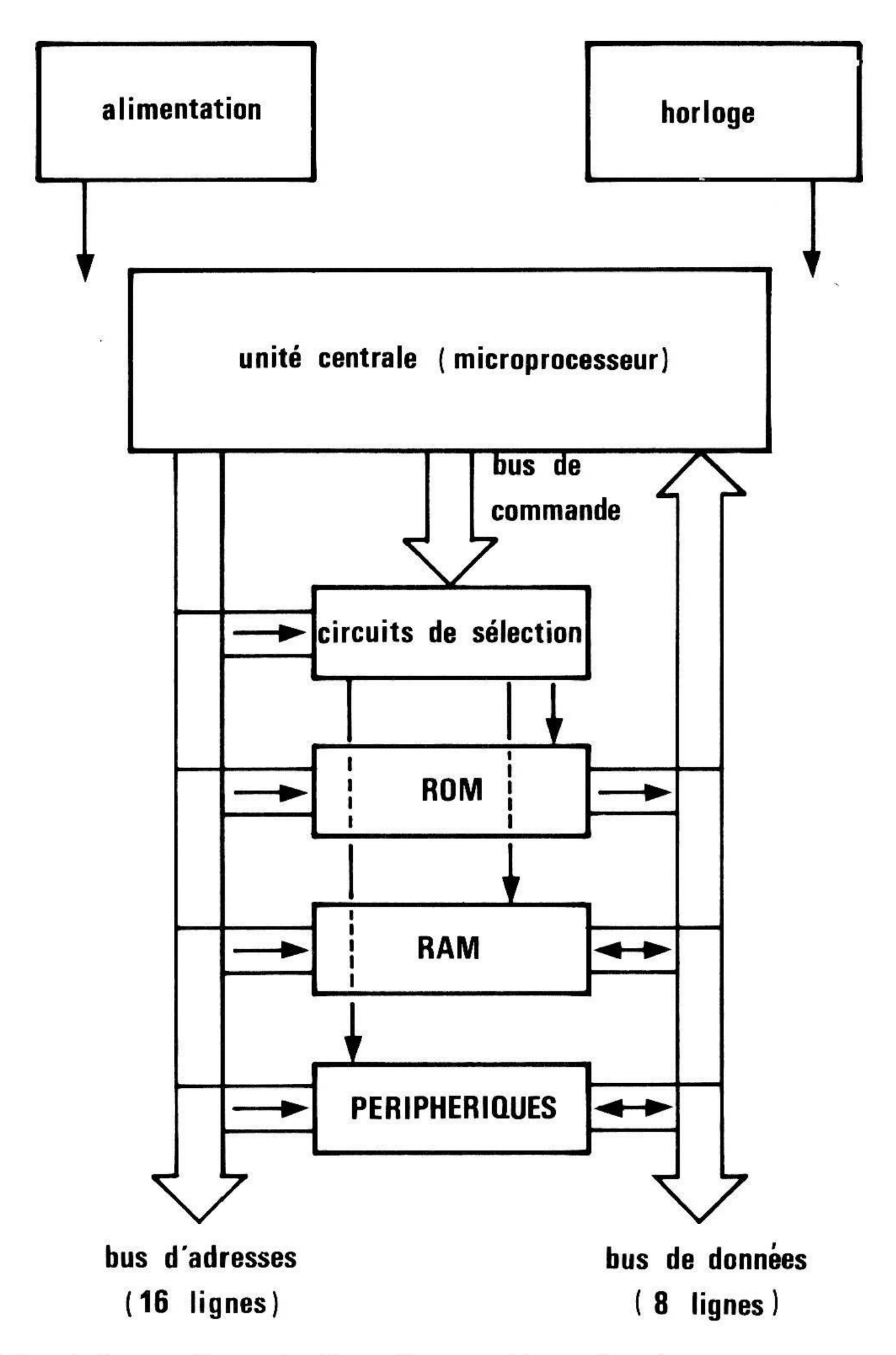


Fig. 1-1. — *Organisation d'un système à microprocesseur.*

Certains petits systèmes ne possèdent pas de RAM, et se contentent des registres. Le ZX 81 de base, lui, possède 1 K-octet de RAM, soit l'équivalent de 1 024 registres.

La mémoire vive est essentiellement *volatile* : tout son contenu s'efface dès la coupure de l'alimentation, mais en revanche, on peut aussi bien y lire qu'y écrire des informations.

• **les périphériques** sont tous les organes qui, connectés à l'unité centrale, servent à communiquer avec l'extérieur : écran vidéo, clavier, interface cassette, imprimante, cartes d'entrée-sortie, cartes sonores, poignées de jeu, etc. Ces « accessoires » jouent un rôle essentiel car, sans eux, rien ne permettrait d'agir sur le microprocesseur qui, lui-même, ne pourrait pas davantage commander quoi que ce soit !

Des liaisons doivent bien sûr exister entre tous ces éléments. Les informations sont constituées de huit *états logiques* (1 ou 0) correspondant à la présence ou à l'absence d'une tension de cinq volts.

Huit fils (plus une masse) devraient donc relier chaque circuit extérieur à l'unité centrale.

Il faut également que le microprocesseur, lorsqu'il échange des informations avec, disons, une mémoire, puisse indiquer à laquelle des « cases » de cette mémoire il s'intéresse : la ROM du ZX 81 comporte 8 192 « cases » ou « adresses », et sa RAM peut en compter 1 024 ou beaucoup plus si des extensions lui sont ajoutées.

En fait, le Z 80 peut distinguer jusqu'à 65 536 adresses différentes (64 fois 1 024 soit 64 K adresses).

Moyennant un certain codage (le code binaire), on peut se contenter de seize fils pour véhiculer une adresse reconnaissable sans ambiguïté.

Pour minimiser, autant que faire se peut, les liaisons déjà touffues nécessaires dans un système à microprocesseur, on fait appel au principe du BUS :

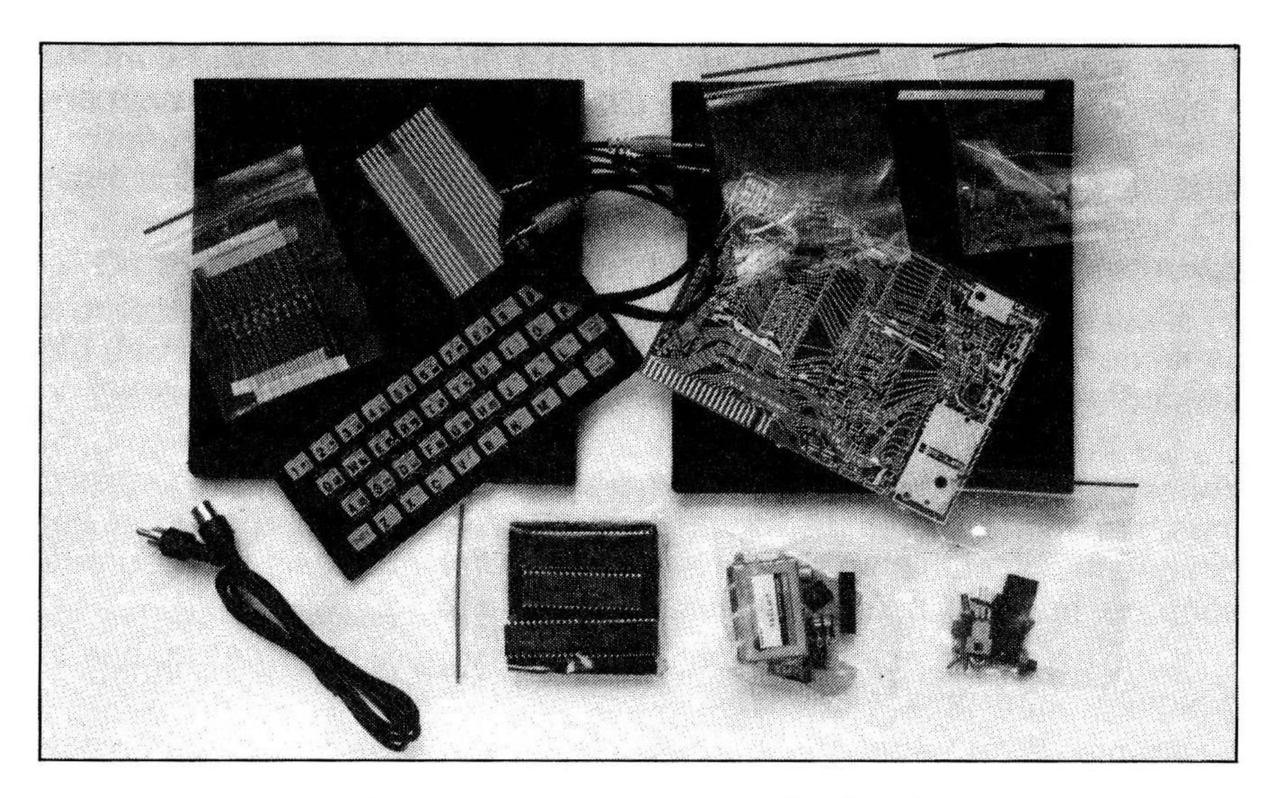
Un BUS est un groupe de lignes desservant, en parallèle, un certain nombre d'organes.

Le Z 80 possède un *bus de données* à huit lignes, et un *bus d'adresses* à seize lignes.

Comme il faut à tout prix éviter un mélange inextricable des données relatives à tel ou tel « abonné » du bus, on utilise le système dit à *trois états* :

Chaque « abonné » (mémoire, périphérique, ou même microprocesseur) est normalement dans un état dit « haute impédance ». Cet état « d'attente » le déconnecte entièrement du bus sur lequel il ne peut ni lire ni écrire.

Des circuits supplémentaires *de sélection* peuvent ordonner à tel ou tel « abonné » de se connecter au bus, soit en tant qu'« émetteur », soit en tant que « récepteur » d'informations. En effet, le bus de données peut véhiculer des informations dans les deux sens : à partir de l'unité centrale ou vers elle.



Le ZX 81 sur la table d'autopsie : une poignée de composants dont l'interconnexion n'est pas toujours dénuée de mystère...

Comme c'est le microprocesseur qui « décide » s'il veut envoyer ou recevoir des données (en fonction des instructions qu'il exécute), il possède deux broches nommées $\overline{RD}$ et $\overline{WR}$. Ces broches, qui font partie du troisième et dernier bus du système, le *bus de commande*, sont normalement maintenues au « 1 » logique (+ 5V).

$\overline{RD}$ passe à zéro lorsque l'unité centrale veut *acquérir* une information. En même temps, le bus d'adresses reçoit les « coordonnées » de l'information à *aller chercher* quelque part sur la « carte mémoire » de 64 K-octets.

Inversement, $\overline{WR}$ passe à zéro lorsque le bus d'adresses véhicule l'adresse à laquelle doit être acheminée une information *venant du* microprocesseur.

Les mémoires mortes, qui ne peuvent qu'être lues, possèdent seulement une entrée « de sélection » : normalement placées dans l'état « haute impédance », elles passent en mode « émission » dès que cette entrée (nommée $\overline{CS}$) est mise à zéro.

Les mémoires vives et les périphériques peuvent fonctionner dans les deux sens : lorsque $\overline{CS}$ est à un, l'état « haute impédance » est forcé. Lorsque $\overline{CS}$ passe à zéro, l'état de l'organe dépend de celui d'une

seconde broche nommée $\overline{WR}$: si $\overline{WR}$ est à un, le mode « émission » ou « lecture » est sélectionné. Dans le cas contraire, on se trouve en mode « réception » ou « écriture ». Il est facile de retenir cela en remarquant que tous les signaux « surlignés » sont *actifs* à l'état zéro, et en sachant que CS signifie Chip Select, WR Write, et RD Read : quelques mots anglais qui reviendront souvent !

Il est évidemment vital (parfois au sens propre en ce qui concerne la santé des composants !) qu'un ordre parfait règne dans les accès au bus de données bidirectionnel.

Où irions-nous, si l'unité centrale envoyait sur le bus une certaine donnée, pendant que la mémoire vive serait en mode « écriture » mais la mémoire morte en mode « lecture » ? Dans le meilleur des cas, une donnée invraisemblable se trouverait écrite en RAM, mais des dégradations de matériel pourraient aussi se produire.

Ce n'est pas directement l'unité centrale qui gère les accès au bus de données, sauf pour elle-même.

Mémoires et périphériques sont placés sous les ordres d'un « chef d'orchestre » nommé *circuit de sélection*.

Ce circuit logique relativement simple (les spécialistes diront *combinatoire* puisqu'il ne comporte que des portes, à l'exclusion de toute bascule, registre, compteur...) *combine* entre eux les signaux $\overline{WR}$ et $\overline{RD}$, et certaines lignes du bus d'adresses unidirectionnel.

On peut affirmer que, mis à part le choix des différentes mémoires et des divers périphériques, c'est le schéma du circuit de sélection qui détermine l'organisation matérielle du système complet, ou *plan de mémoire*.

En plus de ces signaux, le Z 80 génère également une ligne $\overline{MREQ}$, qui passe à zéro quand l'unité centrale cherche à dialoguer avec de la *mémoire* (Memory Request).

Parallèlement, il existe aussi un signal $\overline{IORQ}$ passant à zéro losque le dialogue doit s'établir avec un périphérique d'entrée-sortie (Input Output Request).

En fait, de larges recouvrements peuvent s'opérer entre ces deux modes d'échanges de données.

Le cas du ZX 81 est un parfait exemple de ce que des concepteurs habiles peuvent obtenir par un choix avisé des circuits de sélection : le principal avantage de ces choix est bien sûr une extrême simplicité du schéma (quatre circuits intégrés, parfois cinq), et donc un prix de vente record, avec le succès que l'on sait...

Les sélections mémoire du ZX 81

Les circuits de sélection mémoire du ZX 81 sont inclus, comme bien d'autres fonctions de cette machine, dans le fameux et bien mystérieux « circuit Sinclair ».

On peut cependant se faire une idée assez précise de certains des circuits qu'il contient, en examinant le schéma de cet « ancêtre » du ZX 81 que fut le ZX 80.

Le circuit de cette machine utilisait en effet uniquement des boîtiers TTL courants reliés entre eux par les pistes du circuit imprimé.

C'est ainsi, par exemple, que nous avons reconstitué le schéma des circuits de sélection mémoire, reproduit à la *figure 1-2.*

En fait, c'est purement et simplement la ligne A14 du bus d'adresses qui aiguille le microprocesseur vers la ROM ou la RAM.

En incorporant l'effet de $\overline{\text{MREQ}}$, on aboutit donc à la table de vérité de la *figure 1-3*, qui révèle un « effet secondaire » intéressant : quel que soit l'état de la ligne $\overline{\text{WR}}$, le passage à zéro simultané de $\overline{\text{MREQ}}$ et de A14 entraîne une lecture de la ROM. Or, cette combinaison se présente non seulement lors de l'exécution d'instructions PEEK sur le contenu de la ROM, mais aussi lors de tentatives de POKE, vouées à l'échec puisqu'on ne peut écrire en ROM, mais que rien n'empêche de programmer, ne fût-ce que par erreur.

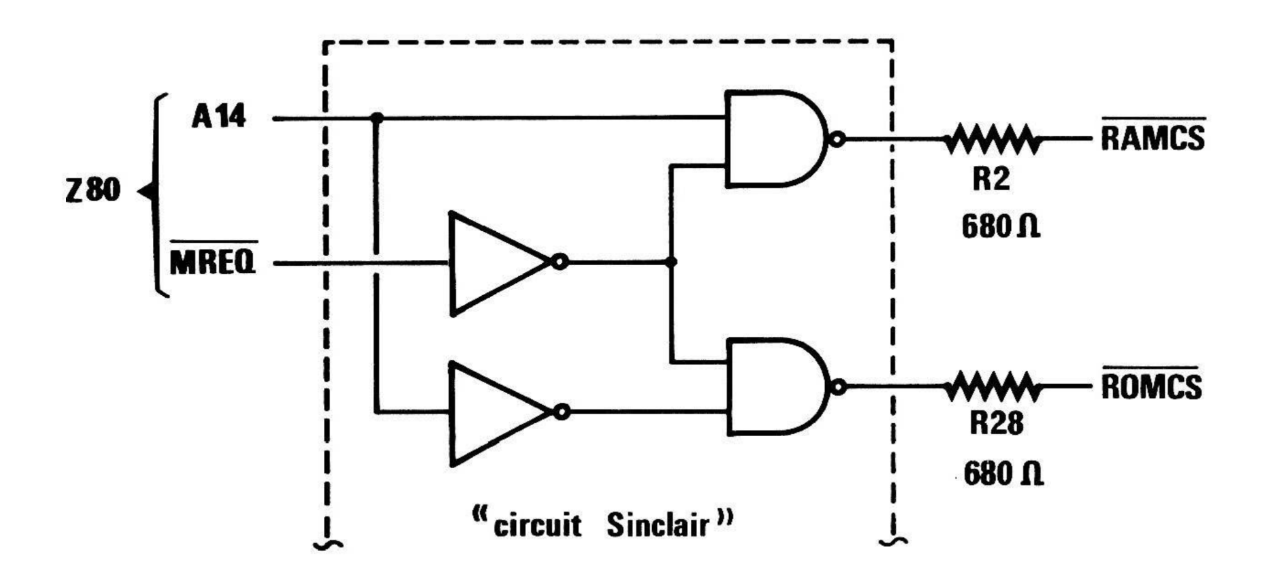


Fig. 1-2.

$\overline{MREQ}$	A14	$\overline{WR}$	Action
0	0	0	lecture en ROM
0	0	1	lecture en ROM
0	1	0	écriture en RAM
0	1	1	lecture en RAM
1	0	0	
1	0	1	mémoires
1	1	0	au repos
1	1	1	

Fig. 1-3.

Dans ces conditions, l'unité centrale enverrait une donnée sur le bus (l'argument de POKE), mais la ROM, en mode lecture, en enverrait une autre, très certainement différente ! Non seulement une telle opération est absurde, mais elle risquerait d'endommager le Z 80 ou la ROM.

C'est certainement pour cette raison que les ingénieurs de chez Sinclair ont incorporé huit résistances en série dans le bus de données (R7 à R15, de 470 ohms).

Par contre, le circuit Sinclair accède directement aux lignes de données de l'unité centrale, avec laquelle il peut donc dialoguer en toute indépendance vis-à-vis des mémoires : ceci a son importance au sein du très complexe processus d'affichage de l'image vidéo.

Du côté du bus d'adresses, un artifice semblable est utilisé, comme en témoignent les *figures 1-4 et 1-5* (brochages de la ROM et de la RAM), et la *figure 1-6* (raccordement des bus au microprocesseur).

La ROM reçoit les lignes AØ à A8 à travers des résistances de 1 kilo-ohm (R22 à R26). Ces lignes « résistives » sont notées A'Ø à A'8, et rejoignent également le circuit Sinclair.

Les lignes A9 à A12 du Z 80 rejoignent directement la ROM, mais ne desservent pas le circuit Sinclair.

En revanche, la RAM reçoit directement de l'unité centrale les lignes AØ à A9 du bus d'adresses.

Tout cela signifie que le circuit Sinclair peut assigner une adresse à la ROM pendant que le Z 80 dialogue avec une adresse de la RAM.

Cette particularité est à la base du fonctionnement de l'affichage vidéo : en effet, la génération du signal TV comporte des tâches trop

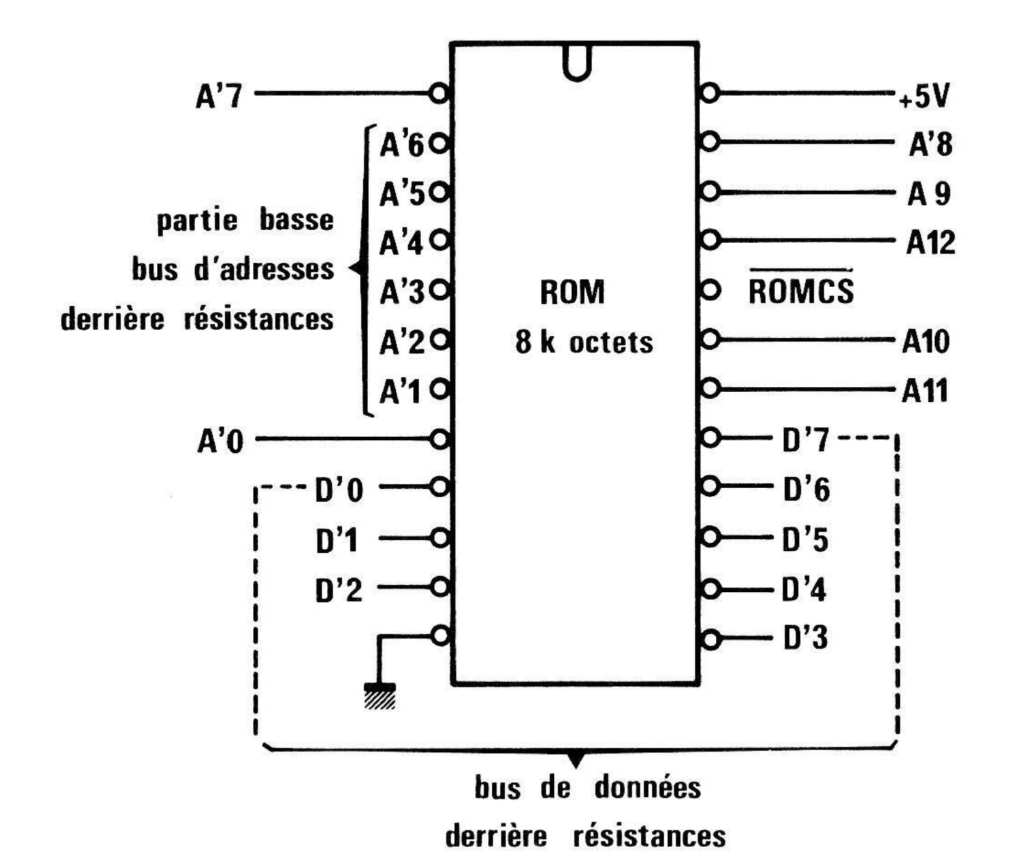

A'7
A'6
A'5
A'4
A'3
A'2
A'1
A'0
partie basse
bus d'adresses
derrière résistances
ROM
8 k octets
+5V
A'8
A 9
A12
ROMCS
A10
A11
D'7
D'6
D'5
D'4
D'3
D'0
D'1
D'2
bus de données
derrière résistances

Fig. 1-4.

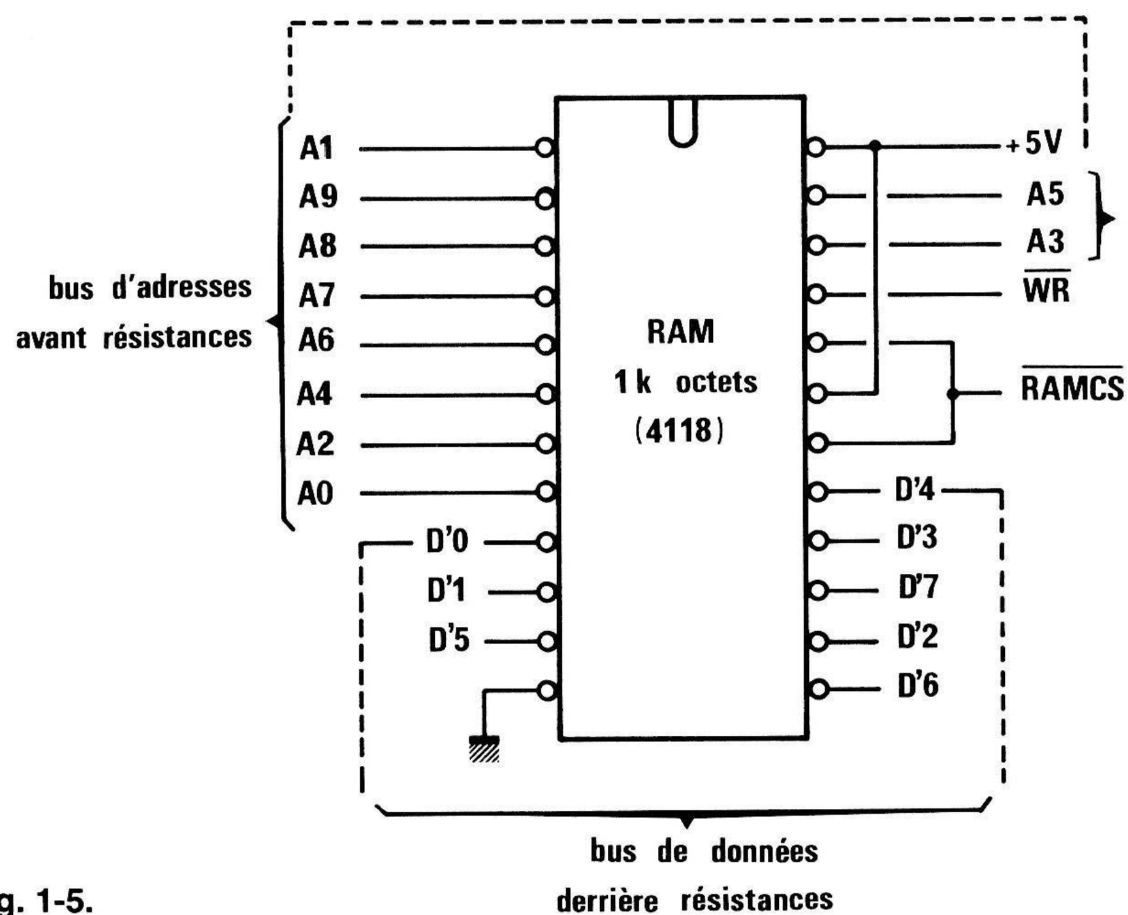
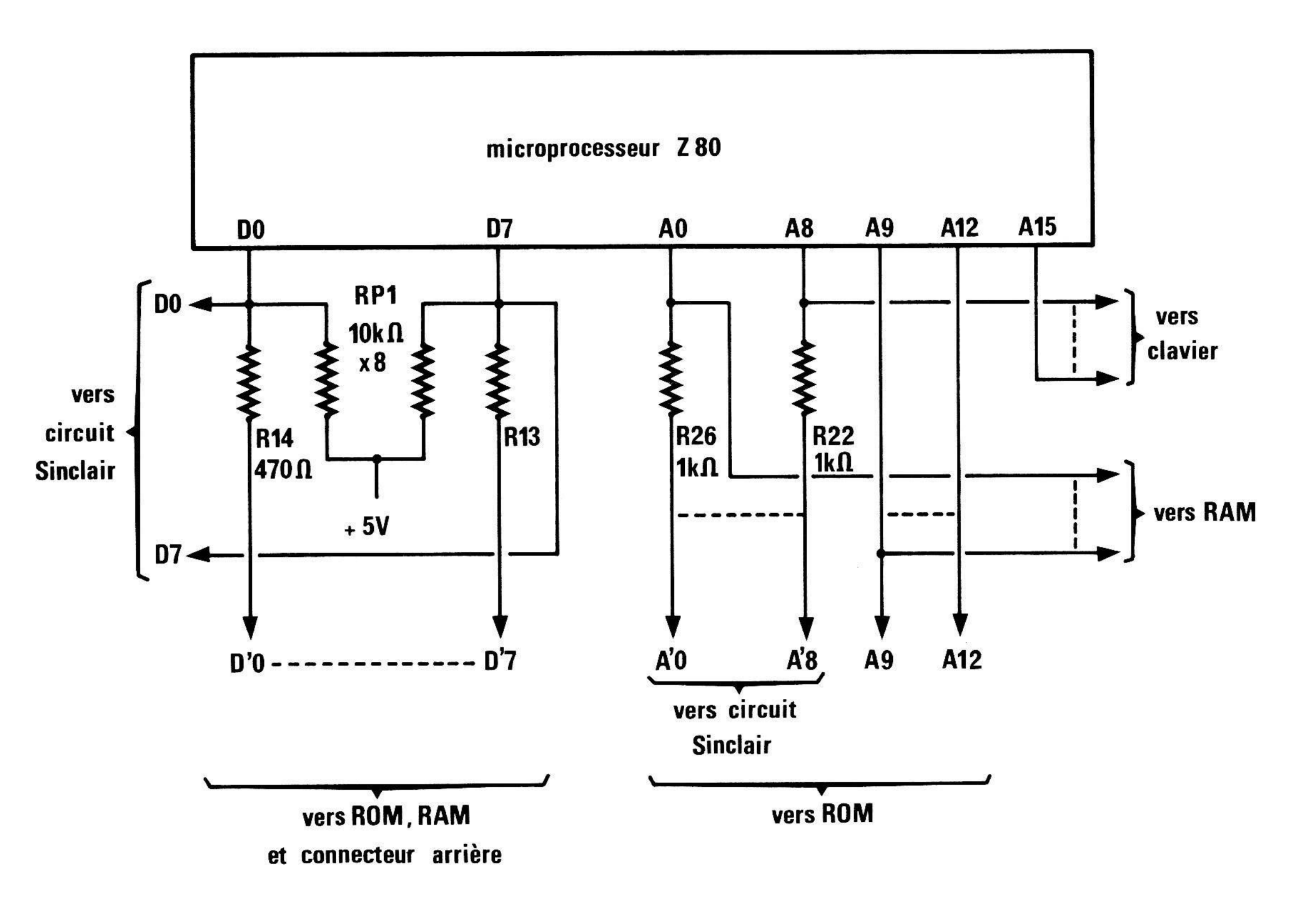

A1
A9
A8
A7
A6
A4
A2
A0
bus d'adresses
avant résistances
RAM
1 k octets
(4118)
+5V
A5
A3
WR
RAMCS
D'4
D'3
D'7
D'2
D'6
D'0
D'1
D'5
bus de données
derrière résistances

Fig. 1-5.


microprocesseur Z 80
D0
D7
A0
A8
A9
A12
A15
D0
RP1
10kΩ
x 8
vers clavier
vers circuit Sinclair
R14
470Ω
R13
R26
1kΩ
R22
1kΩ
vers RAM
+ 5V
D7
D'0
D'7
A'0
A'8
A9
A12
vers circuit Sinclair
vers ROM, RAM
et connecteur arrière
vers ROM
Fig. 1-6.

rapides pour le microprocesseur, et c'est le circuit Sinclair qui s'en charge, notamment en interrogeant le *générateur de caractères* situé en haut de la ROM. Pendant ce temps, le Z 80 exécute des tâches plus lentes, en liaison avec la RAM...

Donc, la ROM reçoit les lignes AØ à A12, avec ou sans résistances : chacune de ces lignes pouvant prendre deux états distincts (0 ou 1), c'est un total de $2^{13} = 8\,192$ combinaisons que l'on peut mettre en évidence à ce niveau.

Ce nombre de combinaisons distinctes du bus d'adresses correspond « comme par hasard » à la capacité de la ROM, 8 K-octets, c'est-à-dire 8 192 cellules (8 × 1 024).

Il est donc clair que, pour peu que le signal $\overline{\text{ROMCS}}$ soit mis à zéro, l'unité centrale pourra lire individuellement n'importe quelle adresse de la ROM.

Seulement, $\overline{\text{ROMCS}}$ ne tient compte que de l'état de la ligne A14 en plus de l'inévitable $\overline{\text{MREQ}}$. Comme il existe deux grands segments de 16 K-octets de l'espace mémoire pour lesquels A14 est à zéro (voir *figure 1-7*), il est clair que la ROM ne pourra faire aucune différence entre, par exemple, l'adresse 8 191 et l'adresse 40 959.

Pour ces deux adresses, en effet, les lignes AØ à A12 ont exactement les mêmes états, et A14 est à zéro.

A l'intérieur d'un même segment de 16 K-octets, la ROM ne tiendra pas davantage compte de l'état de la ligne A13.

Finalement, donc, c'est quatre fois que l'on retrouvera le contenu de la ROM sur la carte mémoire de la *figure 1-7* : une première fois « au bon endroit » (de 0 à 8 191) lorsque A15 et A13 sont à zéro, une seconde fois juste à la suite entre 8 192 et 16 383 (A15 à zéro, mais A13 à un), et deux autres fois de façon symétrique avec A15 à un.

La RAM de base de 1 K-octet subit un traitement similaire : équipée des lignes d'adresses AØ à A9, elle peut distinguer $2^{10} = 1\,024$ adresses différentes, mais ignore superbement les lignes A10 à A13, ainsi que A15.

Résultat de ce dédain, c'est trente-deux fois, pas moins, que ce petit segment de 1 024 octets est « dédoublé » dans l'espace mémoire.

Le gaspillage est considérable : sur 64 K-octets disponibles, pas une adresse ne reste libre alors que seulement 9 K de mémoire sont implantés (8 K de ROM et 1 K de RAM) !

En fait, cette libéralité ne nuit en rien au fonctionnement de la machine de base, mais évite le recours à de complexes (donc coûteux) circuits de sélection.

A15 32768	A14 16384	A13 8192	A12 4096	A11 2048	A10 1024	Carte mémoire
1	1	1	1	1	1	65535
					0	« fantôme » RAM (pour l'affichage)
				0	1	
					0	
			0	1	1	« réflexion » de la partie basse de la carte mémoire
					0	
				0	1	
					0	
		0	1	1	1	
					0	
				0	1	
					0	
			0	1	1	
					0	
				0	1	
					0	49152
	0	1	1	1	1	49151
					0	
				0	1	
					0	« fantôme » ROM
			0	1	1	
					0	
				0	1	
					0	40960
		0	1	1	1	40959
					0	
				0	1	
					0	« fantôme » ROM
			0	1	1	
					0	
				0	1	
					0	32768
0	1	1	1	1	1	32767
					0	
				0	1	
					0	
			0	1	1	
					0	15 « fantômes » de la RAM 1K ou RAM 16 K
				0	1	
					0	
		0	1	1	1	
					0	
				0	1	
					0	
			0	1	1	
					0	RAM 1K
				0	1	
					0	
	0	1	1	1	1	
					0	
				0	1	
					0	« Fantôme » ROM Sinclair (8192 à 16383)
			0	1	1	
					0	
				0	1	
					0	
		0	1	1	1	
					0	
				0	1	
					0	ROM Sinclair 8K (0 à 8191)
			0	1	1	
					0	
				0	1	
					0	

Fig. 1-7.

Un ordinateur conçu dans le strict respect des « règles de l'art » aurait coûté deux à trois fois plus cher, et aurait manqué le succès sans précédent qui fut celui du ZX 81...

C'est lorsque l'on cherche à adapter des extensions (de mémoire ou autres) que les choses se compliquent, car il faut bien dégager de l'espace à la force du poignet !

Fort heureusement, Sinclair a renvoyé les lignes $\overline{ROMCS}$ et $\overline{RAMCS}$ sur le connecteur arrière, derrière leurs résistances de 680 ohms. C'est dire que, de l'extérieur, on peut « forcer » un niveau zéro ou un sur les mémoires, sans risquer d'endommager le microprocesseur qui, lui, n'en fait qu'à sa tête. Bien évidemment, ces forçages ne doivent avoir lieu que dans des conditions parfaitement définies.

Le cas le plus simple est celui de l'extension 16 K-octets Sinclair : le câblage interne du module relie purement et simplement la ligne $\overline{RAMCS}$ au + 5V, bloquant définitivement le bloc d'origine de 1 K-octet de RAM.

Dans l'extension, on trouve 16 K-octets de RAM dynamique, ainsi qu'un tout nouveau circuit de sélection « fabriquant » un signal $\overline{RAMCS}$ adapté à la configuration 16 K RAM.

1 024 octets restent donc « au chômage », mais il existe des solutions pour leur trouver un emploi !

Les extensions de RAM de plus de 16 K-octets doivent éliminer aussi les « fantômes » de la ROM, en agissant, de façon sélective, sur $\overline{ROMCS}$. Des précautions sont à prévoir au-delà de l'adresse 49 151 (extensions 64 K), car les « fantômes supérieurs » de la RAM jouent un rôle dans le processus d'affichage. Des restrictions d'emploi sont donc à prévoir tout en haut de la RAM.

Les accessoires autres que les extensions de RAM (cartes haute résolution, générateurs de caractères, cartes sonores, etc.) affectionnent davantage les emplacements correspondant à des fantômes de la ROM, et tout spécialement au premier (entre 8 192 et 16 383). Il est très facile de libérer cette zone au moyen d'un simple transistor imposant un niveau haut à la ligne $\overline{ROMCS}$ lorsque A13 est à un.

Un montage analogue à celui de la *figure 1-8* se retrouve ainsi dans bon nombre d'accessoires pour le ZX 81.

Le petit programme de la *figure 1-9* permet, pour sa part, de prendre conscience de l'existence des trois « fantômes » de la ROM, voire de vérifier leur présence (ou leur absence) lors de l'adaptation d'accessoires.

Cette possibilité d'accès externe à $\overline{ROMCS}$ et à $\overline{RAMCS}$ constitue véritablement la base de l'évolutivité du ZX 81, puisqu'elle reporte sur les éventuels accessoires externes la complexité (donc le coût) des

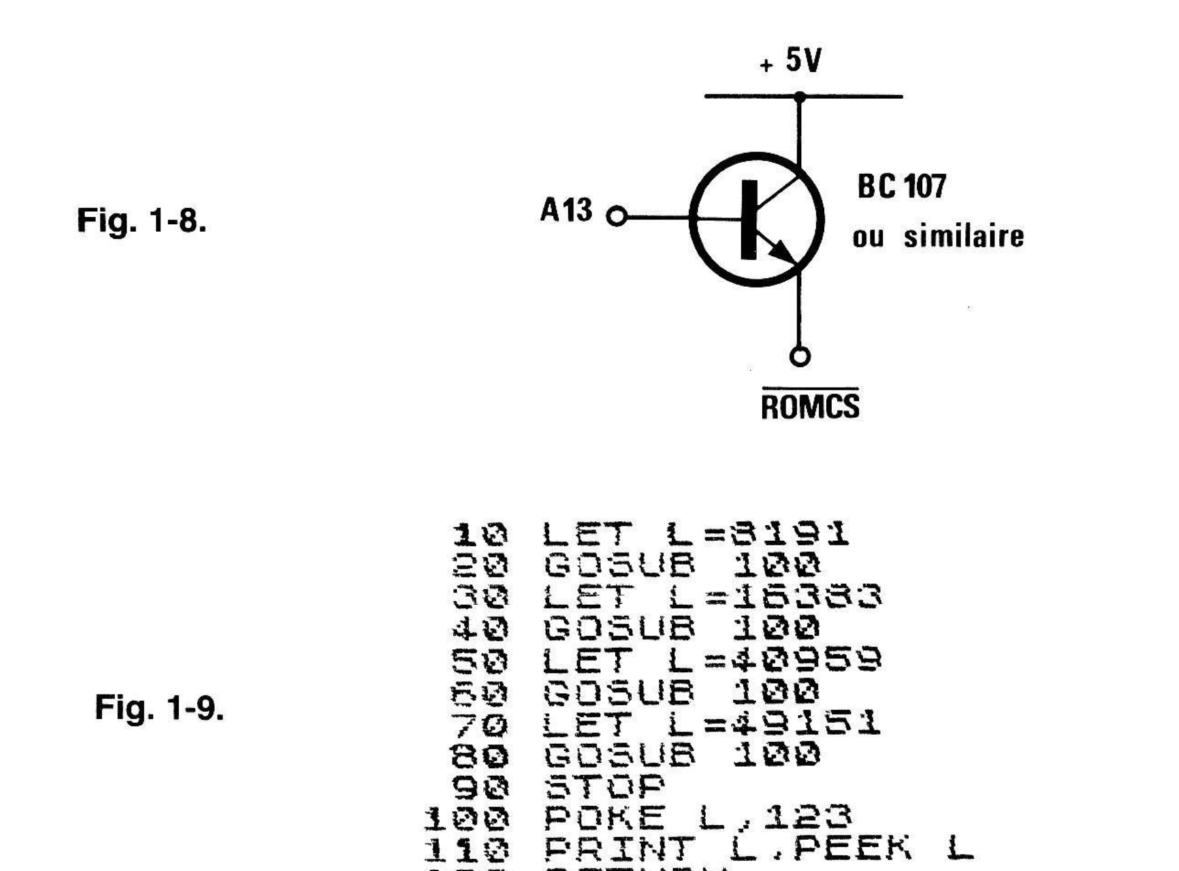


Fig. 1-8.

```
 10 LET L=8191
 20 GOSUB 100
 30 LET L=16383
 40 GOSUB 100
 50 LET L=40959
 60 GOSUB 100
 70 LET L=49151
 80 GOSUB 100
 90 STOP
100 POKE L,123
110 PRINT L,PEEK L
120 RETURN
130 REM COPYRIGHT 1983
```

Fig. 1-9.

sélections spéciales nécessaires, dont l'acheteur de la seule machine de base n'a finalement nul besoin.

Le connecteur arrière de l'ordinateur regroupe tous les signaux de l'unité centrale (y compris les bus complets), plus certaines lignes spécifiques. C'est dire que les rêves les plus fous sont pratiquement tous permis...

Les ports d'entrée-sortie

Nous avons vu que l'unité centrale ne peut dialoguer avec la mémoire (ROM ou RAM) que lorsque la ligne $\overline{MREQ}$ est à zéro.

De façon tout à fait symétrique, l'unité centrale Z 80 peut dialoguer avec 65 536 autres « adresses » lorsque la ligne $\overline{IORQ}$ est à zéro. On ne parle alors plus de cellules mémoire, mais de PORTS d'entrée-sortie.

Il importe de noter que ces « ports » n'ont aucune existence matérielle tant que des circuits appropriés ne sont pas ajoutés au microprocesseur, pas plus qu'une adresse mémoire ne pourrait être utilisée hors de la présence d'au moins un boîtier de ROM ou de RAM.

En fait, la plupart des instructions d'accès aux ports n'utilisent que la moitié inférieure du bus d'adresses, ce qui limite à 256 le nombre de ports couramment employés.

Les périphériques d'origine du ZX 81 (clavier, écran, interface cassette et imprimante) utilisent déjà quatre de ces ports, numérotés 251, 253, 254 et 255 (ou FB, FD, FE, FF en hexadécimal).

Seulement, le décodage de sélection est aussi rustique, sinon plus, que celui des sélections mémoire : l'imprimante, par exemple, utilise le port FB, mais répond en fait à toute mise à zéro simultanée de A2 et de $\overline{IORQ}$. Les ports desservant le générateur d'interruptions destiné à l'affichage, le clavier, et l'interface cassette ne sont guère mieux lotis.

Le résultat est que seuls seize ports restent disponibles à la discrétion de l'utilisateur. Il s'agit des ports 7, 15, 23, 31, 39, 47, 55, 63, 71, 79, 87, 95, 103, 111, 119 et 127.

Bien des possibilités sont offertes par ce nombre, même réduit, d'accès d'entrée-sortie : souvent, un seul port s'avère même suffisant.

Pour utiliser l'un de ces ports libres, il n'est pas nécessaire, comme dans le cas d'une adresse mémoire, d'améliorer le décodage : un simple circuit combinatoire doit reconnaître la combinaison adéquate sur les lignes A0 à A7, ainsi que l'état zéro de la ligne $\overline{IORQ}$.

L'opération d'entrée sera déclenchée par la venue à zéro de la ligne $\overline{RD}$, alors que c'est $\overline{WR}$ qui passe à zéro lorsque le bus contient la donnée que l'unité centrale veut sortir sur le port.

Il est important de remarquer que les échanges d'informations entre le bus et le « milieu extérieur » sont extrêmement brefs : quelques centaines de nanosecondes (moins d'un millionnième de seconde). En fait, les transferts de données ne durent que le temps pendant lequel $\overline{WR}$ ou $\overline{RD}$ sont à zéro. Comme par ailleurs un même port peut aussi bien servir à l'entrée ou à la sortie de données, il est clair que les informations doivent, au niveau du périphérique, être stockées dans des sortes de « registres ».

Le milieu extérieur viendra écrire ou lire dans ces registres à son rythme propre, alors que le microprocesseur fera de même en toute indépendance.

En conséquence, il ne faut pas imaginer qu'il suffit d'appliquer une information sur un port pour que l'unité centrale en prenne aussitôt connaissance : il faut attendre qu'une lecture du port en question se produise, et c'est uniquement le programme en cours d'exécution qui en donne l'ordre.

Cette particularité est spécialement nette en ce qui concerne le clavier : lorsque la machine exécute un programme, le clavier devient sans effet, touche BREAK exceptée, tant qu'une instruction INPUT ou INKEY$ n'est pas exécutée.

En langage machine, il faut prévoir explicitement un sous-programme de lecture du clavier, faute de quoi même le BREAK sera inopérant, avec tout ce que cela signifie...

En effet, c'est par interrogation du port FE (254 en décimal) qu'est effectuée la lecture du clavier.

Le clavier du ZX 81

Elément essentiel du faible coût du ZX 81, le clavier « à effleurement » est constitué de sortes de « circuits imprimés souples » séparés par une feuille souple et isolante percée d'un trou en regard de chaque touche.

Lorsqu'une pression du doigt est exercée sur une touche, les pastilles conductrices des deux circuits se déforment suffisamment pour venir se toucher à travers le trou. Il s'agit donc d'un contact électrique pur et simple, comme dans une touche mécanique conventionnelle.

Les quarante touches du clavier n'utilisent que treize fils pour rejoindre les circuits de l'ordinateur, grâce à une organisation en *matrice* dont la *figure 1-10* donne le détail.



Le clavier du ZX 81 ne manque pas d'ennemis, mais c'est à lui que l'on doit en bonne partie le faible prix de cette petite machine...

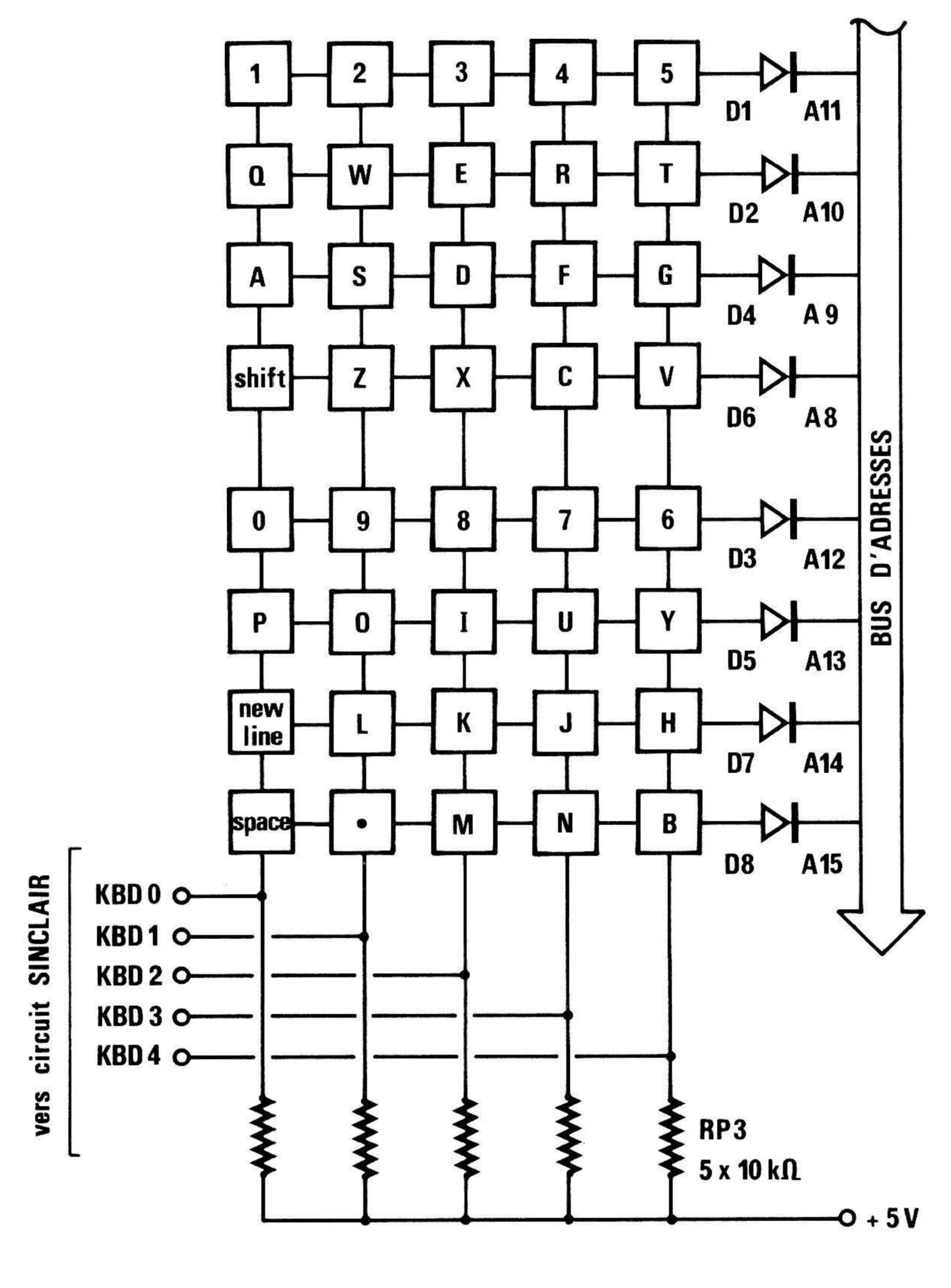


Fig. 1-10.

Sous réserve de bien respecter ce schéma, rien n'empêche de connecter un clavier à touches mécaniques à la place de ou en parallèle avec celui d'origine.

Il est par contre plus délicat de construire un clavier pouvant être branché, comme tant d'autres accessoires, sur le connecteur arrière de l'ordinateur.

En effet, si le raccordement au bus d'adresses ne pose guère de problème (insertion de huit diodes genre 1N4148), la liaison avec les broches KBDØ à KBD4 du circuit Sinclair n'est pas possible.

Ces broches sont en fait les entrées d'un circuit de port, incorporé dans le circuit Sinclair : à chaque interrogation du clavier, le microprocesseur amène tour à tour à zéro chacune des lignes A11 à A15, tandis qu'il interroge le port. Selon la ligne de ce port qui passe à zéro, l'unité centrale peut déterminer laquelle des cinq touches de la rangée commutée est enfoncée.

Pour reconstituer ce fonctionnement à l'extérieur de la machine, il faudrait réaliser un circuit de port d'entrée qui, relié aux bus de données, d'adresses, et de commandes, prendrait la place de celui incorporé dans le circuit Sinclair.

L'opération inverse est d'ailleurs plus facile : brancher des contacts extérieurs en parallèle sur le clavier, que l'on pourra interroger par de simples instructions INKEY$ (réalisation de poignées de jeu, par exemple).

Les interruptions du ZX 81

Nous avons vu qu'une information arrivant sur un port n'est prise en compte par l'unité centrale que selon son bon plaisir, parfois pas du tout si le programme n'a pas prévu une lecture de ce port au bon moment.

Or, certaines nouvelles de l'extérieur doivent être prises en considération sans délai : elles sont *prioritaires*.

Par exemple, les lignes TV doivent être générées à une cadence parfaitement régulière, faute de quoi les circuits du récepteur ne pourraient se synchroniser : une image instable apparaîtrait.

Un processus de génération vidéo doit donc démarrer dès réception d'une impulsion de synchronisation, quitte à *interrompre* provisoirement l'exécution de tâches moins pressantes.

Tous les microprocesseurs acceptent des INTERRUPTIONS : sur réception d'un signal externe, ils suspendent immédiatement le travail en cours, vont exécuter un sous-programme dit de *traitement de l'interruption*, puis reviennent au programme principal là où ils l'avaient laissé. Ainsi, un périphérique peut réquisitionner l'unité centrale dès qu'il en a un besoin urgent.

Seulement, la tâche principale peut parfois être encore plus importante que l'interruption. Il existe donc deux types d'interruptions :

Les *interruptions masquables* ne sont prises en compte qu'avec l'accord du programme principal : deux instructions machine spéciales permettent de spécifier les segments du programme qui sont prioritaires, et ceux qui ne le sont pas.

Les *interruptions non masquables* sont toujours exécutées, quelle que soit la tâche en cours. Elles bénéficient du plus haut degré de priorité possible.

Le microprocesseur Z 80 possède de nombreux modes de traitement des interruptions, utilisés de façon ingénieuse par le ZX 81, notamment en ce qui concerne l'affichage vidéo. Le circuit Sinclair contient un *générateur d'interruptions* pilotant la broche $\overline{\text{NMI}}$ du Z 80 (demande d'interruption non masquable), à une cadence égale à celle de succession des images sur l'écran TV (25 images par seconde).

Dès la mise à zéro de $\overline{\text{NMI}}$, le microprocesseur interrompt la tâche en cours, et part exécuter la routine contenue, dans la ROM, à partir de l'adresse décimale 102. Entre différentes actions de préparation à l'affichage, ce court sous-programme exécute une instruction $\overline{\text{HALT}}$.

Cette instruction met la broche $\overline{\text{HALT}}$ à zéro, ce qui se répercute sur la broche $\overline{\text{WAIT}}$ par l'intermédiaire du transistor TR 1 *(fig. 1-11).*

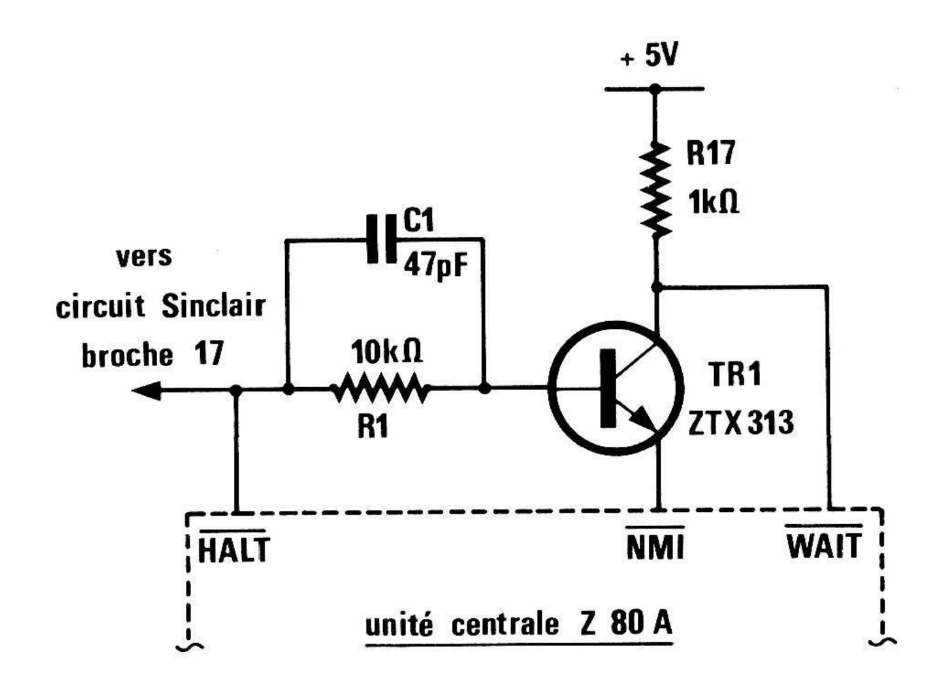


Fig. 1-11.

L'unité centrale attend alors une interruption masquable, réclamée par le circuit Sinclair (dont la broche 10 rejoint l'entrée $\overline{MI}$ du Z 80, entrée de demande d'interruption masquable).

C'est cette routine d'interruption masquable, logée en ROM à partir de l'adresse décimale 56, qui appellera les divers sous-programmes nécessaires à la création d'une image TV, mais seulement si le ZX 81 est en mode SLOW.

En effet, en mode FAST, les interruptions sont neutralisées (le générateur du circuit Sinclair est mis hors service), ce qui multiplie par quatre la vitesse de la machine.

En mode SLOW, en effet, le ZX 81 passe 75 % du temps à s'occuper de l'image TV, contre 25 % seulement à traiter les instructions BASIC...

Il est important de noter que l'appel des routines d'affichage par la routine d'interruption s'opère par une instruction JP (IX).

De ce fait, en modifiant le contenu du registre IX, on peut dévier le microprocesseur vers des routines d'affichage programmées en RAM par l'utilisateur.

L'écriture de telles routines est un travail extrêmement complexe, mais offre des possibilités étonnantes : c'est selon ce procédé que fonctionnent les *logiciels haute résolution graphique*, qui permettent d'accéder, sur l'écran, à 256 × 192 points indépendants sans le moindre accessoire matériel !

La principale leçon à tirer de ce survol des interruptions du ZX 81 est bien entendu qu'un si large usage de leurs possibilités ne laisse guère de place à l'utilisateur de ce côté, sauf, à la rigueur, en mode FAST.

Une très bonne connaissance du microprocesseur Z 80 est alors pratiquement indispensable.

L'interface cassette

Les deux prises destinées au raccordement d'un magnétophone à cassette (MIC et EAR) sont raccordées plus ou moins directement au circuit Sinclair *(fig. 1-12)*. C'est donc lui qui contient les montages transformant les niveaux logiques émanant du microprocesseur en signaux audio, et inversement.

Le circuit débitant sur la prise MIC (micro) est piloté par le port 255 : des effets sonores intéressants peuvent être obtenus en agissant sur ce port au moyen de routines écrites en langage machine. Les signaux ainsi créés pourront être enregistrés ou écoutés par l'intermédiaire du magnétophone, mais atteindront également le récepteur TV, puisque le modulateur vidéo est attaqué par la même broche 16 du circuit Sinclair.

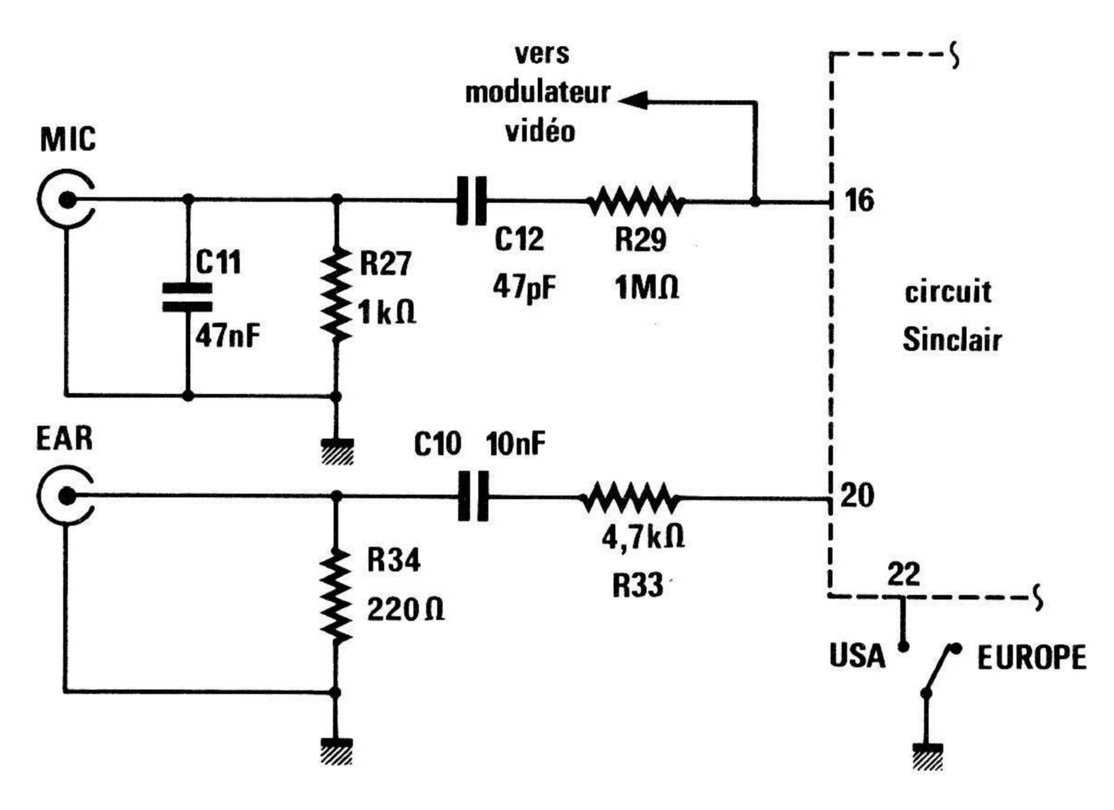


Fig. 1-12.

L'écran affichera donc de curieuses formes, tandis que le haut-parleur émettra certaines sonorités, pourvu que le volume sonore ne soit pas à zéro.

C'est le port 254 qui vient interroger le circuit relié à la prise EAR (écouteur). Mais ce port ne servait-il pas déjà au décodage du clavier ?

En effet, les lignes KBDØ à KBD4 de la matrice du clavier (voir *figure 1-10*) correspondent bien à ce port FE.

Seulement, un port peut accueillir jusqu'à huit lignes : il en reste donc trois, qui ne sauraient être perdues !

La ligne 5 rejoint la broche 22 du circuit Sinclair, et contrôle donc la présence ou l'absence du strap qui constitue le moyen de transformer le ZX 81 en une machine adaptée au standard américain de télévision 525 lignes/60 Hz. La ligne 6 passe à un lorsque des données valides entrent en machine par la prise EAR, alors que la ligne 7 véhicule les données elles-mêmes, sous la forme d'un train de bits série.

Une application possible de cette remarque est la détection de signaux audio quelconques appliqués, avec un niveau suffisant, à la prise EAR. Une simple interrogation périodique du port FE montrera immédiatement si un signal est présent ou non.

De très simples routines en langage machine permettront donc à l'amateur imaginatif d'utiliser les prises cassette très au-delà de leur vocation d'origine.

Les circuits annexes

Même si ces circuits n'offrent pas d'immenses possibilités de recherche à l'amateur curieux, il est bon de connaître le fonctionnement sommaire de l'alimentation, de l'horloge, et de la remise à zéro.

L'alimentation secteur, dont le schéma général apparaît à la *figure 1-13* est scindée en deux parties : un bloc secteur aussi simplifié que possible, qui fournit une tension non régulée variant entre 9 et 15 volts environ.

Des modèles 700 mA et 1,2 A existent, selon que l'ordinateur fonctionne avec ou sans imprimante.

Une régulation à 5 V exactement est opérée dans le ZX 81 lui-même par un classique régulateur « 3 pattes » 7805. Quelques condensateurs de découplage judicieusement répartis aux points stratégiques du circuit complètent le schéma. Comme le 7805 a tendance à chauffer notablement, il peut être bénéfique d'ajouter un montage pré-régulateur entre le bloc secteur et l'ordinateur (par exemple un régulateur 7808) ; on y gagnera en fiabilité.

Le plus important est cependant de noter que cette alimentation travaille déjà non loin de ses limites de sécurité. Si des accessoires

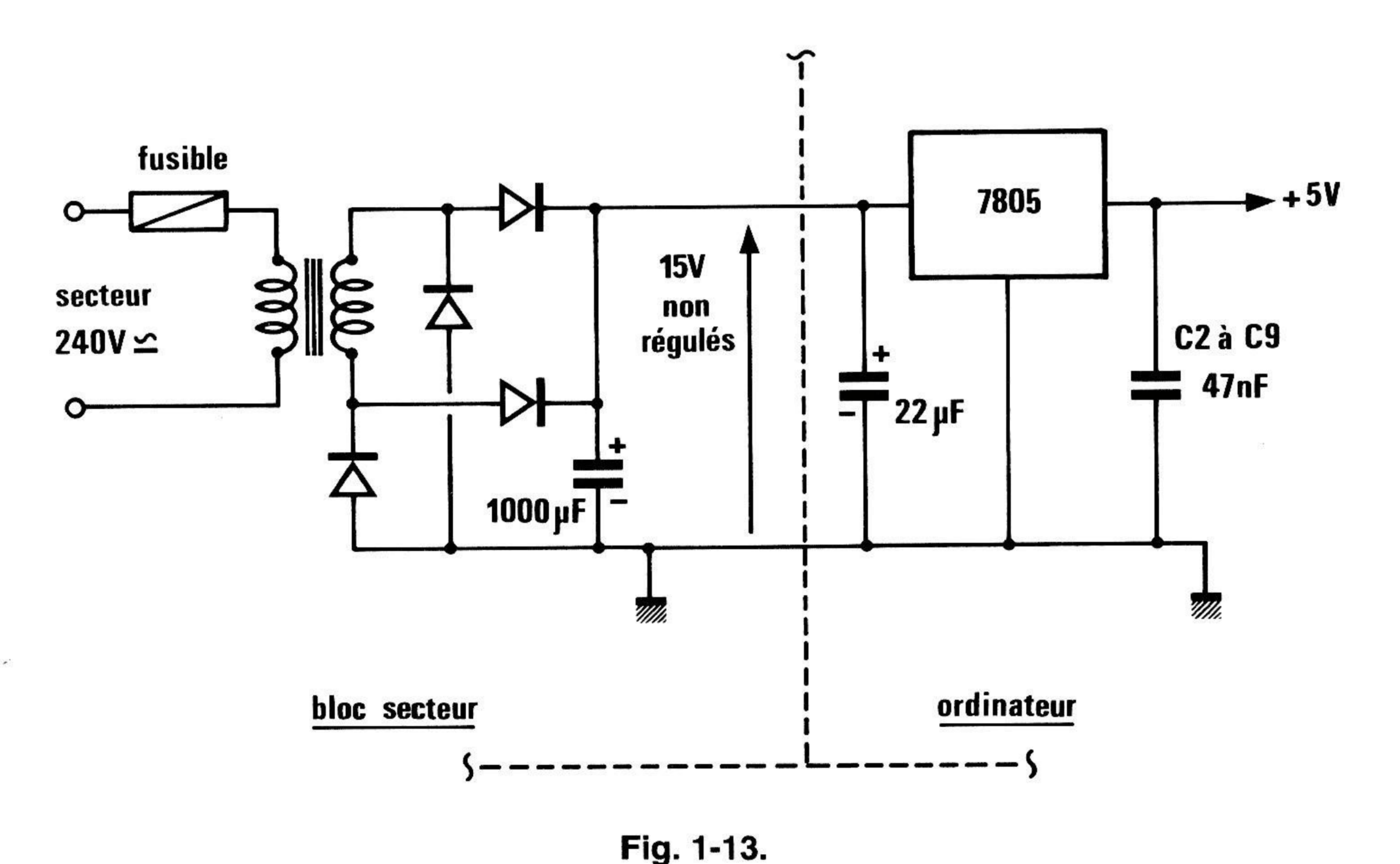


Fig. 1-13.

« gourmands » devaient être adjoints, il pourrait s'avérer utile de les alimenter séparément.

L'oscillateur d'horloge est incorporé dans le circuit Sinclair, mais utilise un élément de référence extérieur. Au lieu d'un quartz, classique à ce niveau, un filtre céramique de téléviseur a été choisi en raison de son

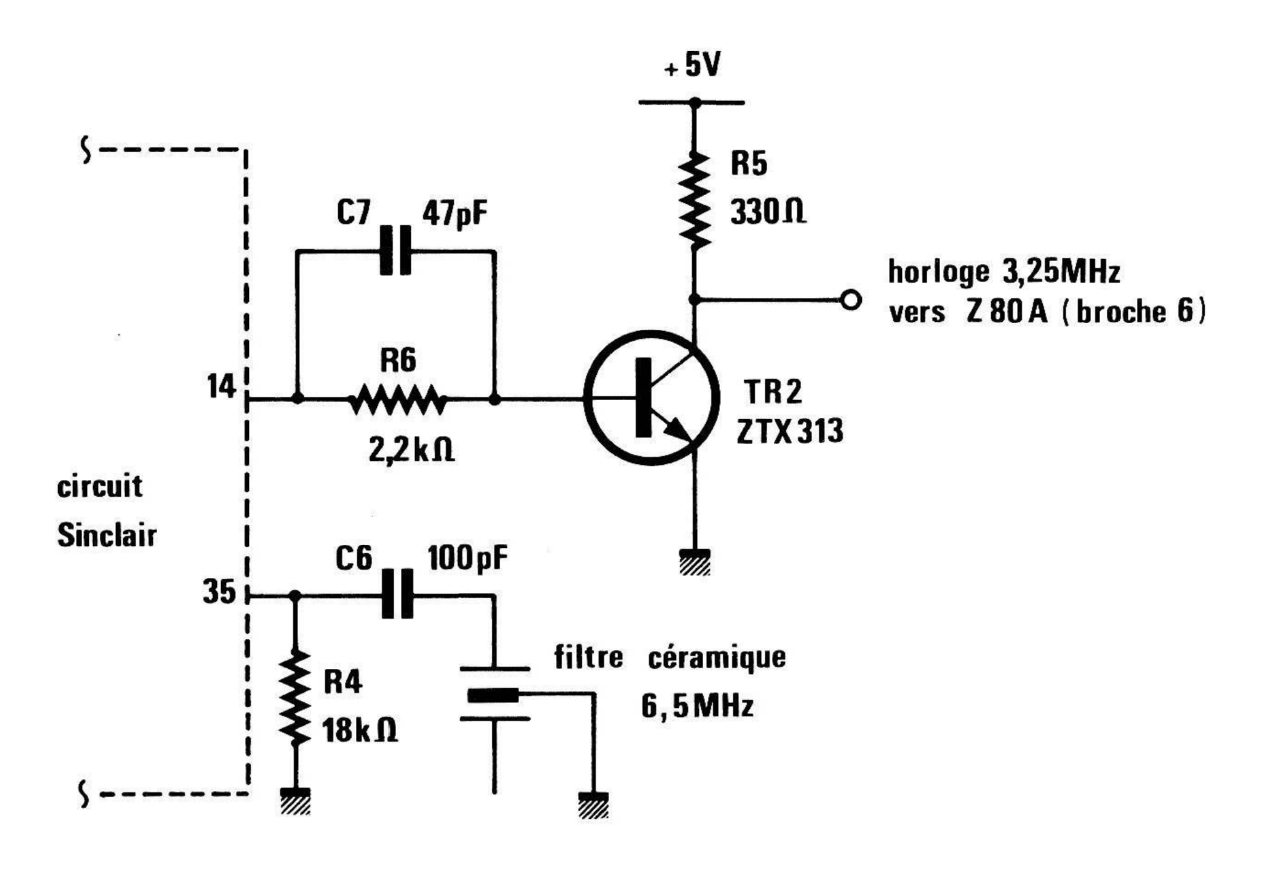


Fig. 1-14.

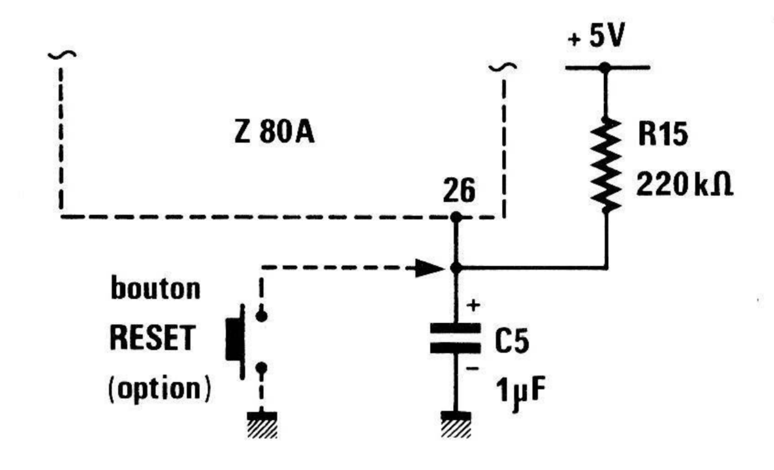
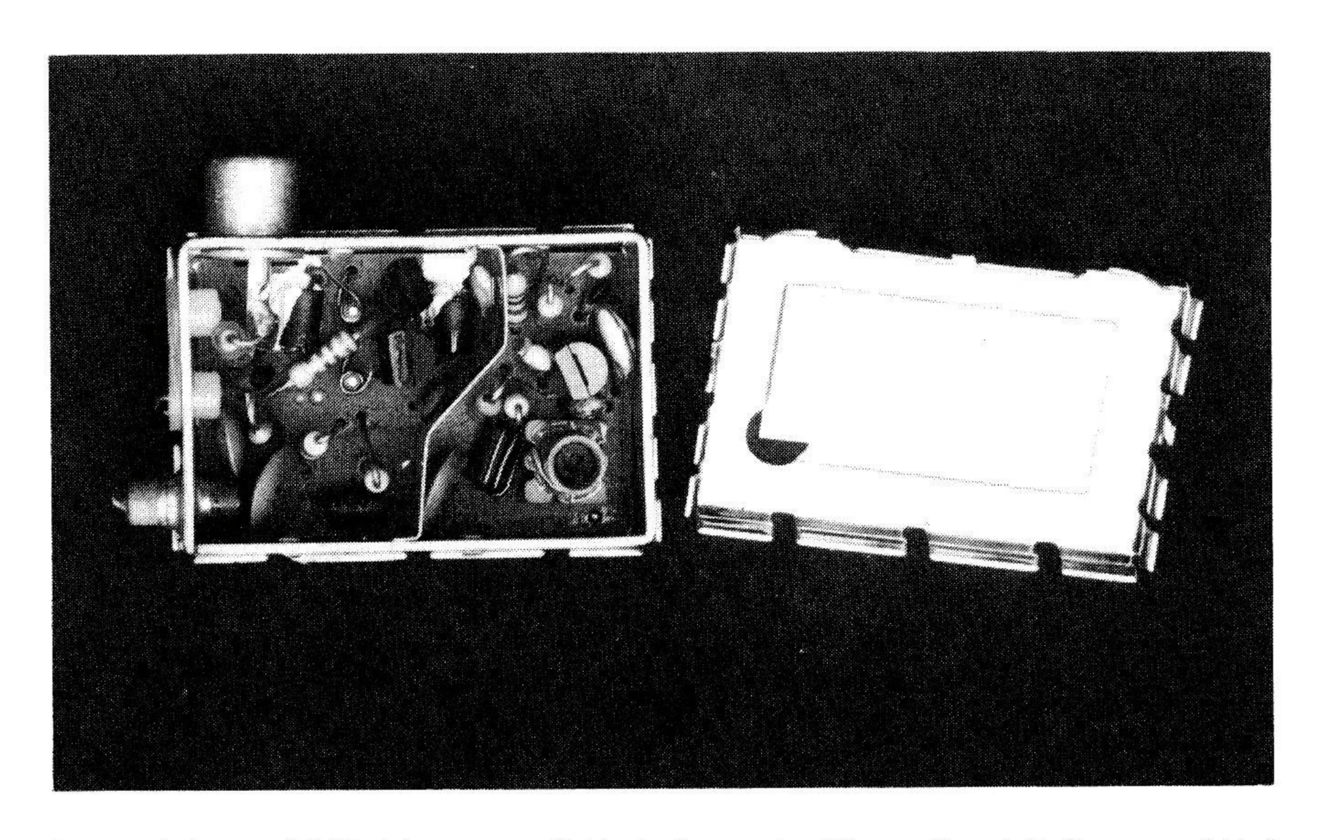


Fig. 1-15.



Le modulateur UHF, bien que câblé de façon traditionnelle, doit être considéré comme un circuit intégré à l'intérieur duquel il n'est guère question d'intervenir.

coût très inférieur. La précision de fréquence est moindre, mais encore suffisante.

La fréquence de l'oscillateur sert directement au fonctionnement des systèmes vidéo du circuit Sinclair (la valeur de 6,5 MHz convenant fort bien à ce niveau).

L'unité centrale reçoit pour sa part un signal d'horloge de 3,25 MHz, obtenu par division à l'intérieur du circuit Sinclair. Un transistor extérieur opère une dernière mise en forme *(figure 1-14)*.

Enfin, le petit circuit de la *figure 1-15* exploite le fait que la mise à la masse, même brève, de la broche 26 du microprocesseur, fait reprendre à celui-ci l'exécution du programme à partir de l'adresse zéro (le début de la ROM). C5 étant déchargé à la mise en service du ZX 81, la machine démarrera ainsi toujours dans les conditions voulues. Un bouton de RESET peut éventuellement être ajouté ici.



Chapitre 2

Une nouvelle vie pour le ZX 81

Le ZX 81 est essentiellement destiné à exécuter des programmes BASIC implantés en RAM, soit à partir d'une frappe au clavier, soit suite à un chargement à partir d'une cassette audio.

Toute l'organisation interne de la machine, que nous avons détaillée dans le précédent chapitre, est optimisée en vue de ce but principal.

Certes, des possibilités d'accès au langage machine sont prévues, mais dans des conditions de commodité pour le moins discutables.

Quoi qu'il en soit, le ZX 81 contient tous les éléments d'un puissant système à microprocesseur, reliés entre eux par des bus complets et entièrement accessibles sur le connecteur arrière.

On peut donc songer à convertir le ZX 81, au moyen de transformations simples, en un véritable automate programmable capable de recevoir des informations de l'extérieur, de les traiter avec toute la puissance de l'outil informatique, puis d'agir en conséquence sur des organes externes.

Bien des systèmes de contrôle industriel utilisent des ressources matérielles nettement plus minces que celles offertes par le ZX 81, aussi peut-on envisager sans honte de confier à cette petite machine la gestion d'un système d'alarme, d'un aquarium, d'un train miniature, et de bien d'autres équipements domestiques dont le mode de fonctionnement se prête à être automatisé.

Cependant, si l'on accepte d'immobiliser ainsi son ZX 81 de façon quasi-permanente, on hésitera sans nul doute à faire de même avec un téléviseur et un magnétophone... Les systèmes industriels à microprocesseur se contentent le plus souvent de quelques voyants pour toute visualisation, et démarrent l'exécution de leur programme dès leur mise sous tension, sans aucun chargement manuel ! La suite de cet ouvrage n'aura pas d'autre but que de décrire les adaptations pratiques à exécuter sur le système ZX 81, pour lui permettre de fonctionner de la sorte.

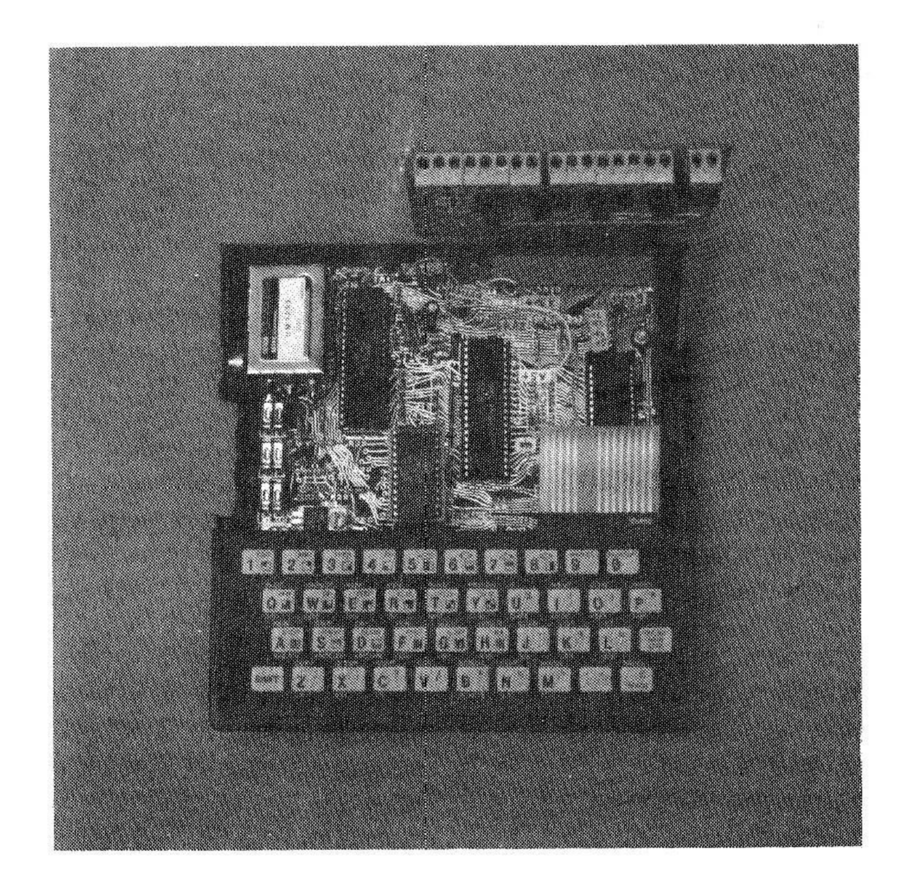
Bien sûr, nous fournirons des exemples concrets de mise en œuvre pratique, mais nous nous attacherons surtout à donner à nos lecteurs tous les moyens leur permettant de résoudre eux-mêmes leur propre cas particulier, ce qui est le principal intérêt d'une solution micro-informatique.

Quelques extensions matérielles

Pour pouvoir communiquer valablement avec le monde extérieur, l'ordinateur doit bien évidemment disposer de moyens adaptés.

On songe immédiatement à ces « cartes d'entrée-sortie » du commerce, parmi lesquelles la « 8ES » est sans nul doute la plus répandue en France.

Nous décrirons bien sûr des applications de ces extensions dont bien des détenteurs de ZX 81 sont déjà équipés, mais nous insisterons aussi sur la *construction* de cartes présentant des caractéristiques différentes.



Les extensions capables de donner une nouvelle vie au ZX 81 se montent soit à l'extérieur, soit dans le boîtier même de l'ordinateur.

En effet, huit entrées et huit sorties ne suffisent souvent pas, alors que pour certaines applications, une seule entrée est nécessaire, voire même pas d'entrée du tout.

Parallèlement à cette mise à la disposition de la machine de moyens d'entrée-sortie, nous aurons à modifier la configuration de sa mémoire, par trop spécialisée dans la programmation BASIC.

Pour ce faire, les lignes $\overline{\text{ROMCS}}$ et $\overline{\text{RAMCS}}$ du connecteur arrière nous seront naturellement d'un grand secours, mais nous n'hésiterons pas à aller beaucoup plus loin, quitte à supprimer carrément la ROM Sinclair pour lui substituer une EPROM, programmable et effaçable à volonté par l'utilisateur.

Le recours à des EPROMs (mémoires mortes reprogrammables et effaçables aux ultra-violets) constitue la base des adaptations que nous ferons subir au ZX 81.

En effet, lorsqu'un programme est « figé » dans ce type de mémoire, il devient immédiatement disponible par simple enfichage dans la machine : plus de cassette, plus de liste à frapper au clavier, mais un simple interrupteur à actionner !

Bien sûr, programmateur et effaceur seront deux outils indispensables, mais nous découvrirons que le ZX 81 peut se transformer temporairement en un très honnête poste de programmation, alors que la

plupart des électriciens vendent de quoi construire un effaceur tout à fait correct.

Lorsque la nostalgie de l'écran TV se fera trop vive, nous découvrirons comment adjoindre à la machine une ligne d'afficheurs alphanumériques qui n'aura rien à envier aux distributeurs de billets les plus modernes.

Si des effets sonores, ou plus techniquement des signaux à fréquence audible, devaient être produits par le système, nous lui adapterions un synthétiseur capable de contenter aussi bien les musiciens que les techniciens de laboratoire.

Enfin, quelque chose nous dit que certains de nos lecteurs ne résisteront pas à l'envie d'automatiser plusieurs fonctions indépendantes, peut-être même d'équiper différents locaux.

Afin de leur éviter l'achat d'un autre, ou de plusieurs autres ZX 81, nous avons mis au point une véritable *carte microprocesseur* dont l'architecture compatible permettra d'utiliser les mêmes programmes.

Des circuits d'alimentation et des étages de puissance permettront ainsi de construire sans limitation des systèmes entièrement autonomes, à partir de composants aisément disponibles dans le commerce.

Quelques procédés de programmation

La programmation des systèmes à microprocesseur souffre d'une tenace réputation de complexité, car elle s'opère traditionnellement en *langage machine.*

Ce mode de programmation pourra bien sûr être utilisé sur les systèmes qui vont être décrits, et nous savons bien que nombre de nos lecteurs ne s'en effaroucheront pas. Pour les autres, nous proposerons un programme BASIC capable d'écrire directement le langage machine à partir des instructions que l'utilisateur frappera au clavier en langage clair.

Enfin, nous découvrirons comment loger directement un programme BASIC dans une EPROM, afin de lui offrir les mêmes possibilités d'appel instantané qui sont celles du langage machine.

C'est dire que toutes les catégories d'utilisateurs du ZX 81 trouveront matière à innover, sans pour autant dépasser les limites de leurs connaissances ou de leur... audace !

Nous savons fort bien que certains de nos lecteurs n'osent pas soulever le capot de leur chère machine. Bien que les risques de détérioration restent très minces pour un bon amateur électronicien, nous fournirons des solutions permettant d'obtenir des résultats intéressants sans intervenir à l'intérieur de l'ordinateur !

Chapitre 3

Des entrées et des sorties



Les dispositifs d'entrée-sortie sont des accessoires d'importance vitale pour toute application d'automatisation par microprocesseur ou micro-ordinateur.

Selon les cas, il faut mettre en œuvre des circuits de sortie, des circuits d'entrée, ou même les deux, en nombre variable.

Egalement, la nature des organes commandés et des « capteurs » change beaucoup d'une application à l'autre.

Pour des raisons purement informatiques (manipulation des variables sous forme d'*octets*), il est commode de prévoir entrées et sorties par blocs de huit.

La carte « 8ES », très répandue chez les utilisateurs de ZX 81, possède justement huit entrées et huit sorties.

La carte 8ES

La carte 8ES est un circuit imprimé de fabrication française qui vient s'enficher sur le connecteur arrière du ZX 81 et se trouve alimenté par la machine.

Il possède trois jeux de robustes bornes à vis, destinées à recevoir les connexions menant aux organes extérieurs. Deux bornes rejoignent la masse générale, qui constitue le point commun à tous les contacts que l'on peut raccorder aux huits entrées, regroupées sur un même bornier.

L'état des huit contacts peut être acquis par l'unité centrale grâce à une simple interrogation du *port* attribué à la carte.

Il s'agit là d'une opération qui ne peut être exécutée que sous langage machine, mais de très simples routines peuvent être facilement incorporées à un programme BASIC (voir la notice de la carte et notre ouvrage *Maîtrisez votre ZX 81*, dans cette même collection).

Selon les versions de la carte 8ES et le choix de l'utilisateur (voir notice), le numéro du port peut varier. **Dans toute la suite de cet ouvrage, nous supposerons que ce port est le n° 127.**

Pour tout autre port, il faudrait bien sûr apporter les très simples correctifs indispensables à nos logiciels.

Les huit sorties de la carte pilotent à la fois des diodes électroluminescentes (petits voyants très commodes pour les essais) et de robustes transistors capables de commander des courants atteignant deux ampères sous trente volts d'alimentation.

En effet, ce n'est pas le ZX 81, déjà bien sollicité, qui alimente les organes commandés (relais, voyants, petits moteurs, électrovannes, etc.).

On prévoiera donc un bloc d'alimentation de caractéristiques adaptées aux organes pilotés par la carte. Le pôle négatif de ce bloc rejoindra la masse, tandis que son pôle positif viendra desservir le point commun de toutes les « charges » utiles.

Enfin, chaque charge sera reliée individuellement à sa borne personnelle, sur le bornier de sortie.

Pour certains types de charges (les spécialistes parleront de charges *réactives*), des protections par diodes ou condensateurs pourront avantageusement être ajoutées.

Notons que l'intérêt d'une telle carte est que l'ordinateur se trouve presque totalement protégé contre les fausses manœuvres électriques qui, commises directement sur son connecteur arrière, risqueraient d'entraîner sa fin tragique. En présence d'une carte 8ES, les éventuels dégâts se limiteraient à quelques composants peu coûteux et faciles à remplacer.

40 entrées en parallèle sur le clavier

Le fonctionnement du clavier a été détaillé dans notre chapitre 1, et nos lecteurs savent facilement « interroger » ses touches en BASIC comme en langage machine.

Il est facile de souder des fils sur le côté soudures du circuit imprimé de la machine, en parallèle sur les deux connecteurs recevant les « queues » du clavier.

L'un de ces connecteurs regroupe les « lignes » de la matrice, l'autre les colonnes. En soudant un fil d'un bouton-poussoir sur le premier et l'autre sur le second, quelles que soient les broches choisies, on disposera d'une entrée commode. La manœuvre peut être renouvelée jusqu'à concurrence de 40 fois, mais avec deux réserves :

— les entrées ainsi obtenues ne pourront être « actives » simultanément : lorsque deux touches du clavier sont enfoncées, seule la première pressée est reconnue,

— les deux fils d'une entrée de ce type doivent être électriquement « flottants » : pas question de relier l'un ou l'autre d'entre eux à la masse !

Ce procédé ne peut donc résoudre tous les problèmes d'entrées, mais offre déjà d'intéressantes possibilités.

Une carte à vingt sorties

Principe général des interfaces de sortie

Les circuits de sortie sont les plus simples des dispositifs d'interface pour micro-ordinateurs. En effet, l'unité centrale accompagne toute modification dans l'état de ses bus, de l'émission de signaux de synchronisation. Il suffit alors de piloter par ces signaux des mémoires de type « latch » pour stocker aussi longtemps que voulu les états essentiellement fugaces des bus.

En général, c'est le bus de données (à huit bits sur les microprocesseurs les plus courants) qui véhicule les informations à sortir sur les « périphériques ».

Le décodage du bus d'adresses et des signaux de synchronisation permet aux circuits de sortie de ne tenir compte que des données qui leur sont strictement réservées. Selon la nature de ce décodage et les procédures logicielles utilisées, les informations sortantes peuvent soit transiter par des « ports », soit utiliser certaines adresses mémoire.

L'avantage de la première solution est de garder intact l'espace mémoire disponible, mais se paie par la nécessité de recourir au langage machine.

Dans le cas du ZX 81, dont l'espace mémoire est très loin de son plein emploi, il est plus facile de recourir au second procédé qui utilise de simples ordres POKE. Bien plus, le choix de certaines adresses mémoire supérieures à 32 K permet de faire passer un « complément d'informations » sur les lignes basses du bus d'adresses. C'est ainsi que nous avons pu obtenir très simplement vingt lignes de sortie : les huit lignes du bus de données, et les douze lignes « basses » du bus d'adresses. Avantage supplémentaire, il est possible d'utiliser tout à fait indépendamment ces deux groupes de bits, comme nous le découvrirons plus loin.

Etude d'une carte à vingt sorties

La *figure 3-1* reproduit le schéma de principe de notre carte, construite autour de très courants SN 7475 (quadruples latches). La micro-informatique procure une nouvelle jeunesse à ces circuits TTL initialement destinés à servir de mémoires d'affichage.

Chaque sortie est équipée d'un transistor en collecteur ouvert permettant toute une variété de branchements directs.

Le cœur du montage réside cependant dans les circuits de décodage pilotant les entrées de commande des 7475. Ces entrées ne recevront l'impulsion d'écriture WR (complément de $\overline{WR}$ du Z 80) que lorsque toutes les conditions suivantes seront remplies à la fois :

— $\overline{MREQ}$ à Ø (c'est-à-dire mémoire sélectionnée, par opposition à une opération sur port, qui mettrait $\overline{IORQ}$ à Ø à la place de $\overline{MREQ}$),

— A12, A13 et A15 à 1 (ce qui correspond à une adresse mémoire supérieure à 45055, donc inutilisable même par une extension 16 K),

— A14 à Ø (afin que la RAM soit bloquée, évitant ainsi toute ambiguïté due au décodage simplifié de $\overline{RAMCS}$ et $\overline{ROMCS}$ dans le ZX).

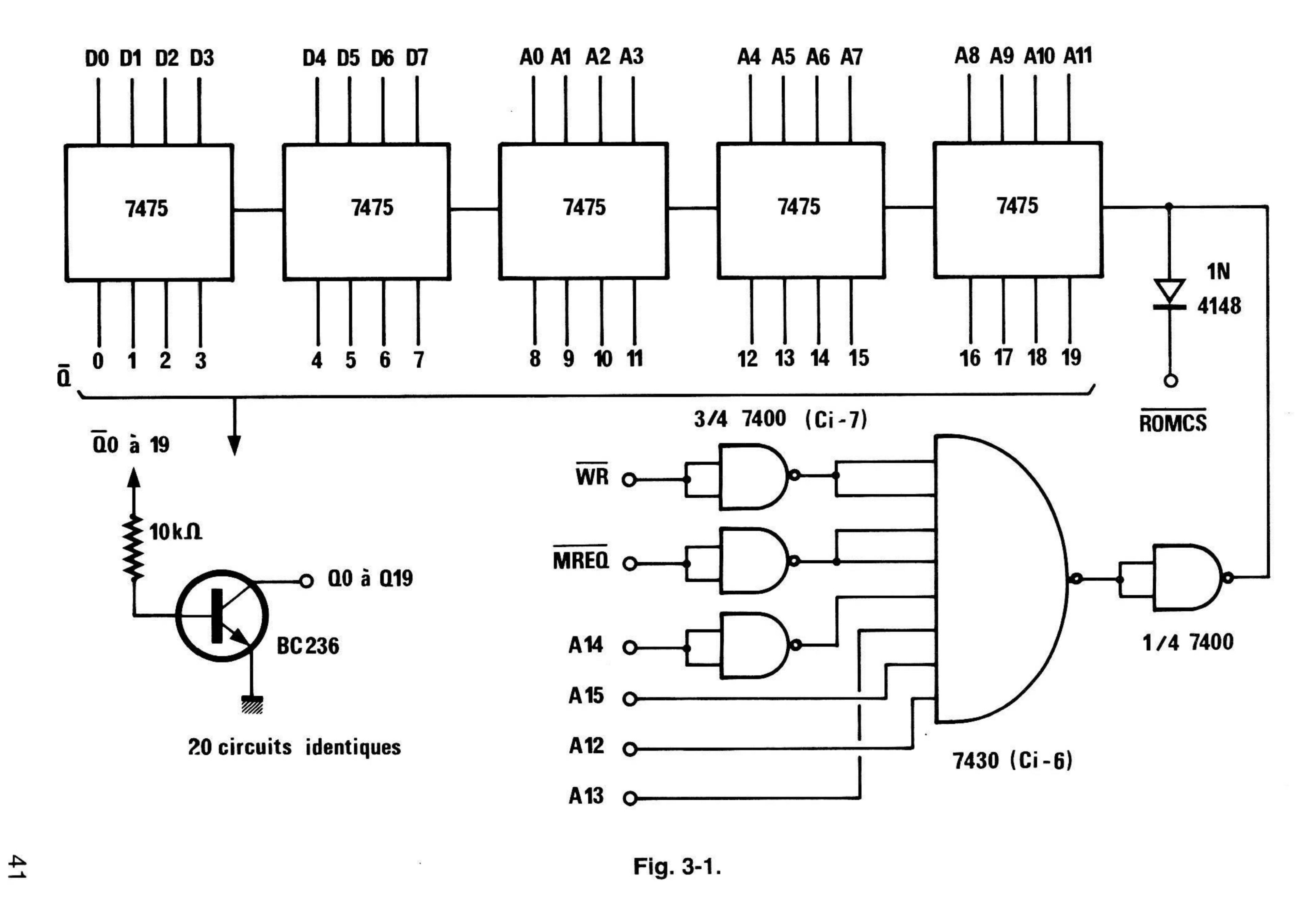

D0 D1 D2 D3
D4 D5 D6 D7
A0 A1 A2 A3
A4 A5 A6 A7
A8 A9 A10 A11
7475
7475
7475
7475
7475
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
$\overline{Q}$
1N 4148
$\overline{ROMCS}$
$\overline{Q}$0 à 19
3/4 7400 (Ci-7)
$\overline{WR}$
$\overline{MREQ}$
A14
A15
A12
A13
1/4 7400
7430 (Ci-6)
10kΩ
Q0 à Q19
BC236
20 circuits identiques

Fig. 3-1.

Cependant, afin d'empêcher que la non-sélection de la RAM n'entraîne la sélection de la ROM, une diode vient forcer à 1 la ligne $\overline{ROMCS}$. La RAM et la ROM étant toutes deux inhibées, les bus se trouvent ainsi réservés à l'usage exclusif de la carte de sortie pour toutes les adresses mémoire comprises entre 45056 et 49151. Notre carte « consomme » donc 4 K-octets d'espace mémoire, ce qui pourrait paraître dispendieux si cette zone n'était laissée à l'abandon par les concepteurs du ZX 81 !

Seule conséquence négative, il ne faudra pas utiliser avec cette carte d'autres modules d'extension de RAM que le 16 K d'origine. Que l'on se rassure cependant, les programmes d'application proposés se contentent de 1 K-octet ! Si maintenant nous lançons un ordre de la forme :

POKE 45056 + X, Y

en présence de la carte, les sorties Ø à 7 de celle-ci prendront la valeur binaire correspondant à Y (via les huit bits du bus de données), alors que les sorties 8 à 19 prendront la valeur binaire de X (via les bits AØ à A11 du bus d'adresses).

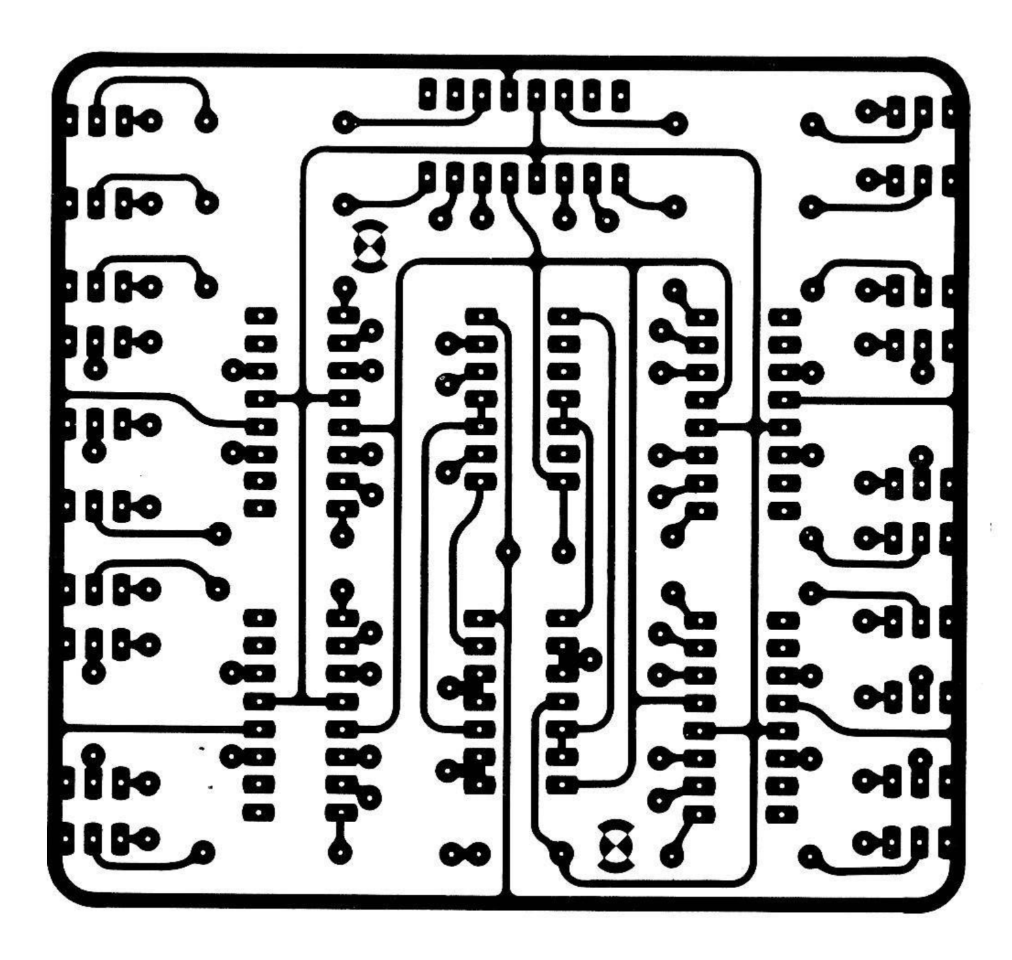


Fig. 3-2. — *Carte « Interface » (voir aussi la figure 6-5).*

Cette possibilité de sortir simultanément un nombre compris entre Ø et 255, et un autre compris entre Ø et 4095 sera vitale pour l'application de synthèse de fréquence qui va être décrite plus loin.

Réalisation pratique

Le circuit imprimé représenté *figure 3-2* a été dessiné en vue de recevoir tous les composants du montage, à l'exception des organes de raccordement à l'ordinateur. On pourra en effet choisir entre deux solutions concurrentes :
— raccordement permanent au moyen d'une nappe de fils directement soudés sur le circuit du ZX 81,
— utilisation d'un connecteur gigogne à 44 broches, enfiché à l'arrière de la machine.

Un autre avantage de cette option est qu'il sera possible, moyennant quelques modifications mineures, d'adapter la carte à d'autres ordinateurs basés sur l'emploi du Z 80. La *figure 3-3* reproduit le plan

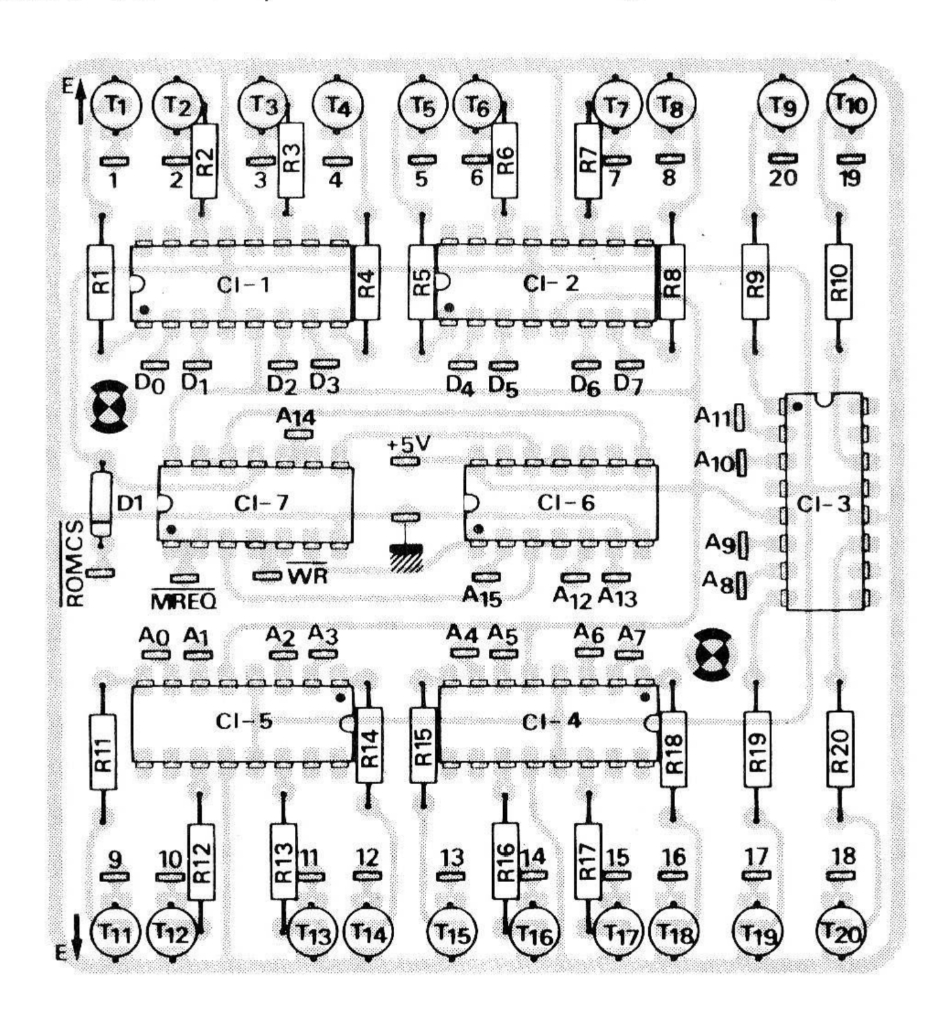
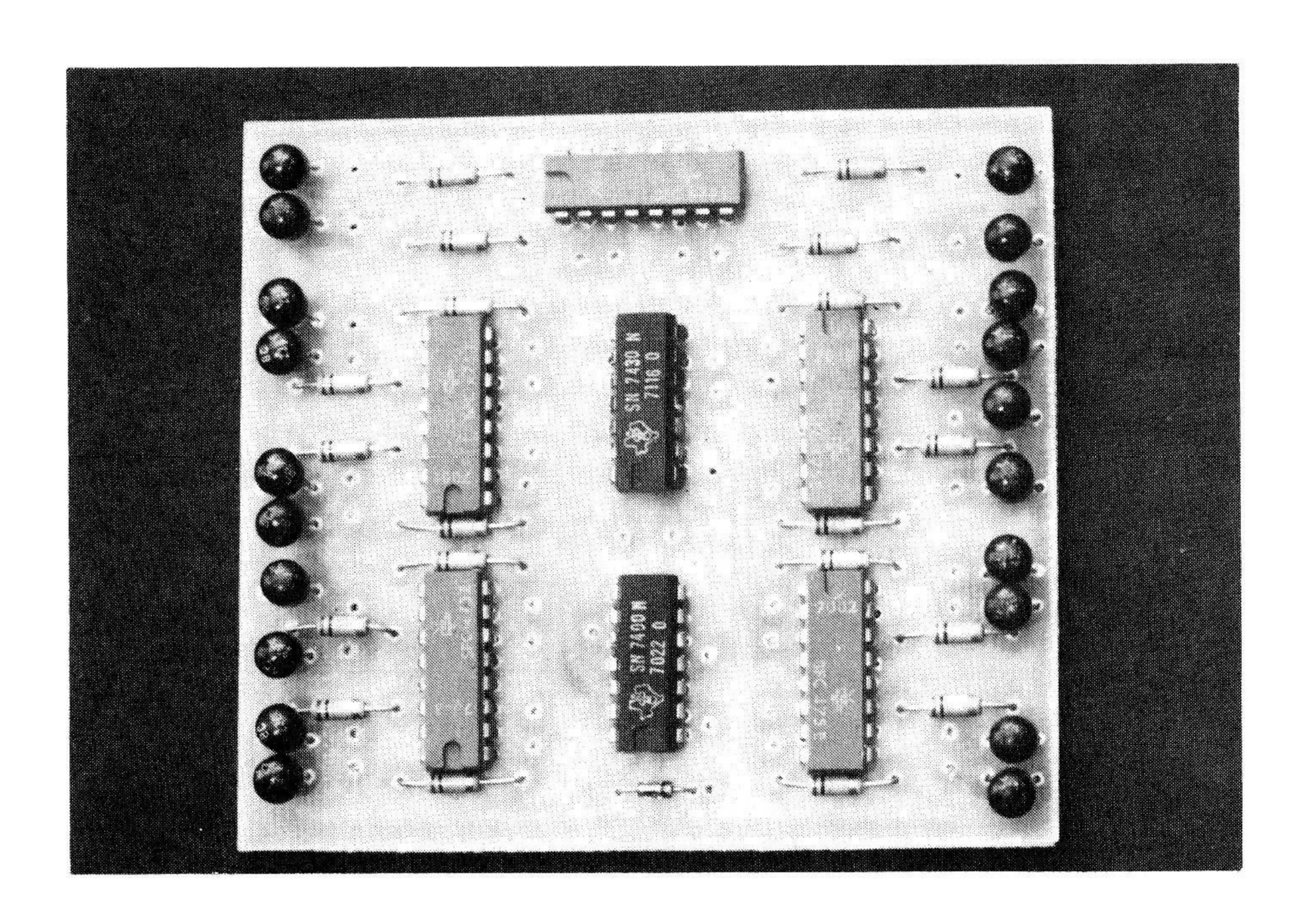


Fig. 3-3. — *(Nature des composants : figure 3-1).*



La carte à vingt sorties prête à être utilisée.

d'implantation, qui ne soulève pas de commentaires particuliers. Insistons seulement sur la nécessité d'une constante vigilance lors du raccordement à l'ordinateur, tout « croisement » de fils pouvant bloquer tout le système.

Chaque fois que la chose sera possible, on utilisera de préférence des circuits intégrés de la famille TTL « LS », qui consomment moins tout en réduisant la sollicitation des bus de la machine. Les références à approvisionner seront donc de la forme 74 LS 75, 74 LS 00, etc.

Pour tester la carte achevée, il suffit de lancer quelques ordres POKE bien choisis, et de contrôler l'effet produit au niveau des sorties, soit au moyen d'un contrôleur, soit grâce à des diodes LED. Par exemple, la commande :

POKE 45056,0

doit mettre tous les collecteurs des transistors de sortie à la masse, alors que :

POKE 49151,255

doit bloquer ces mêmes transistors tous à la fois. Notons qu'il existe *plus*

d'un million de combinaisons possibles, et qu'il ne saurait donc être question de les essayer toutes !

Application : Un synthétiseur HF programmable

Notre but ne sera pas ici de détailler le fonctionnement des synthétiseurs de fréquence, qui a déjà été largement étudié par ailleurs. Nous nous bornerons donc à décrire les principales caractéristiques du montage dont la *figure 3-4* donne le schéma de principe, avant de traiter de son raccordement à notre carte d'interface.

Ce montage rassemble des circuits intégrés appartenant à des familles très diverses, depuis un linéaire jusqu'à un diviseur ECL, en passant par les technologies MOS et CMOS. Le fonctionnement extrêmement performant qui en résulte est le suivant :

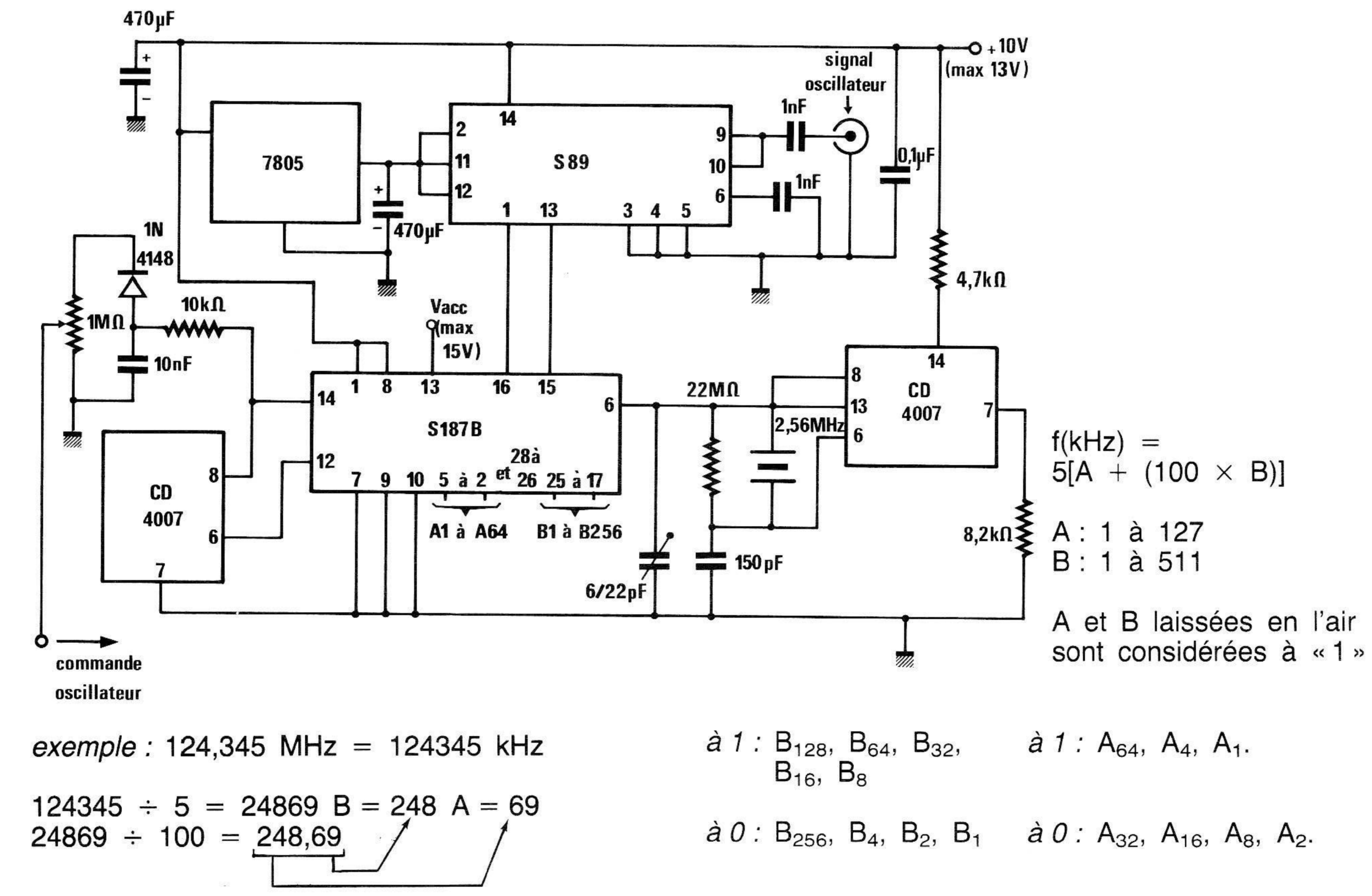
Si le montage est attaqué par une fréquence générée par un oscillateur, et que sa sortie vienne commander une diode varicap insérée dans le circuit accordé de cet oscillateur, on pourra alors asservir avec la précision du quartz la fréquence de cet oscillateur à des informations binaires appliquées aux seize lignes d'entrée du S 187 B. Bien sûr, il faut choisir une fréquence tombant dans le domaine pouvant être couvert par l'oscillateur.

En simplifiant les choses, le synthétiseur délivre tout bonnement une tension d'autant plus positive que la fréquence de l'oscillateur est inférieure à la valeur programmée. Alimentée par cette tension, la varicap agit de façon à rétablir l'égalité.

Pour ce faire, il faut fournir aux diviseurs programmables apparaissant sur la *figure 3-5*, deux nombres A et B, compris respectivement entre 0 et 128, et entre 0 et 512. Dès lors, la fréquence programmée obéit à la relation suivante : $F\ (kHz) = 5(A + (100 \times B))$

On en déduit que l'incrément, ou pas, du synthétiseur sera de 5 kHz, ce qui est un espacement courant entre canaux radiotéléphoniques. L'éventail théorique de fréquences serait de 5 kHz à 256,640 MHz, mais les limitations du prédiviseur S 89 restreignent celui-ci à la plage de 500 kHz à 250 MHz. On peut cependant souvent déborder de ces limites sans problèmes majeurs.

Si l'application de cette formule permet un choix très souple de n'importe quelle fréquence, elle se prête mal à une commande manuelle, par roues codeuses par exemple. C'est là que l'informatique apporte une solution extrêmement élégante : l'ordinateur peut facilement calculer A et B pour n'importe quelle fréquence, et transmettre ces deux valeurs au synthétiseur, par l'intermédiaire des deux parties d'une instruction POKE, et... de la carte d'interface.



f(kHz) =
5[A + (100 × B)]

A : 1 à 127
B : 1 à 511

A et B laissées en l'air sont considérées à « 1 »

exemple : 124,345 MHz = 124345 kHz

124345 ÷ 5 = 24869 B = 248 A = 69
24869 ÷ 100 = 248,69

à 1 : B_{128}, B_{64}, B_{32}, B_{16}, B_8 *à 1 :* A_{64}, A_4, A_1.

à 0 : B_{256}, B_4, B_2, B_1 *à 0 :* A_{32}, A_{16}, A_8, A_2.

Fig. 3-4. — *Synthétiseur universel de fréquence. Synthétiseur 500 kHz à 250MHz au pas de 5kHz.*

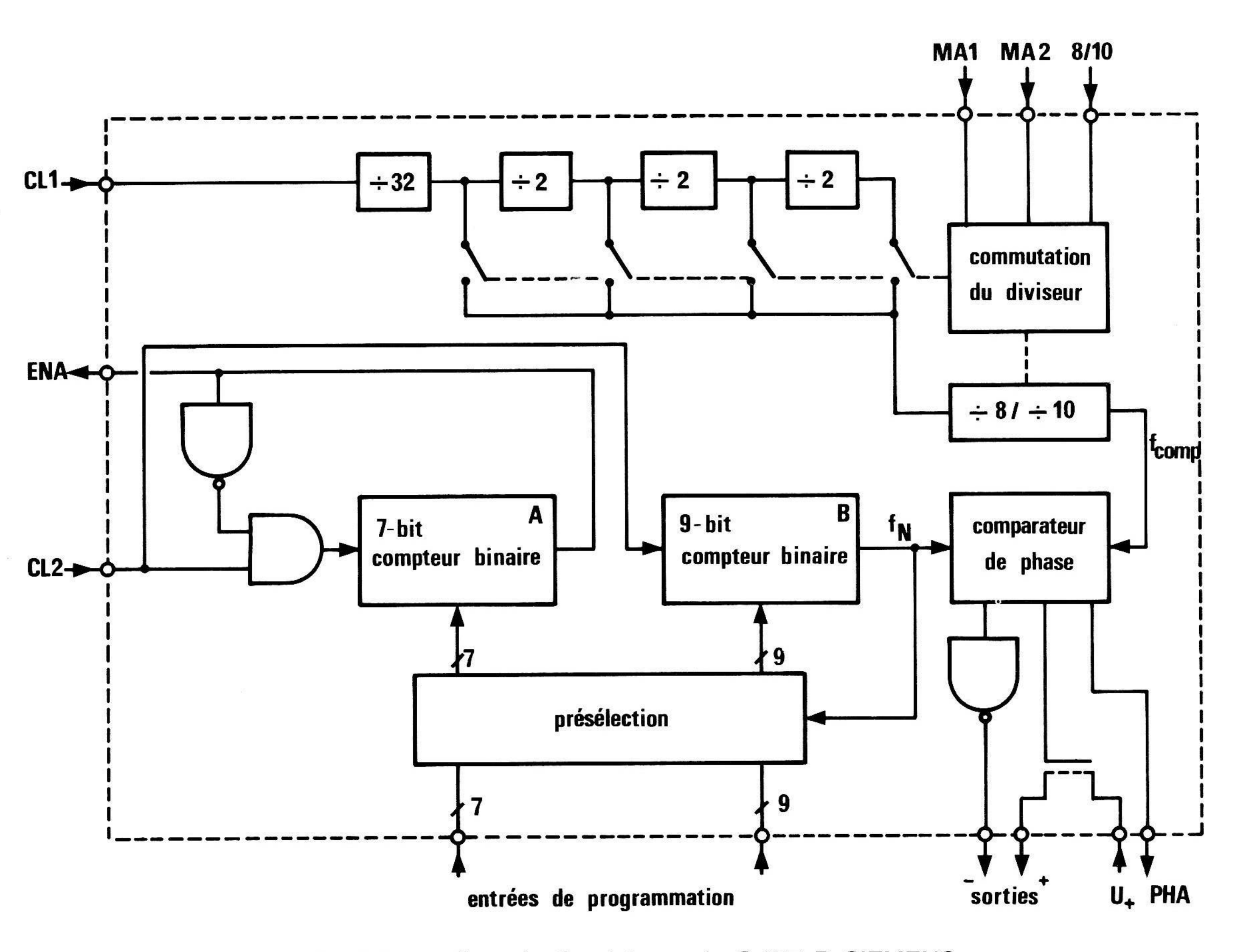


Fig. 3-5. — *Organisation interne du S 187 B SIEMENS.*

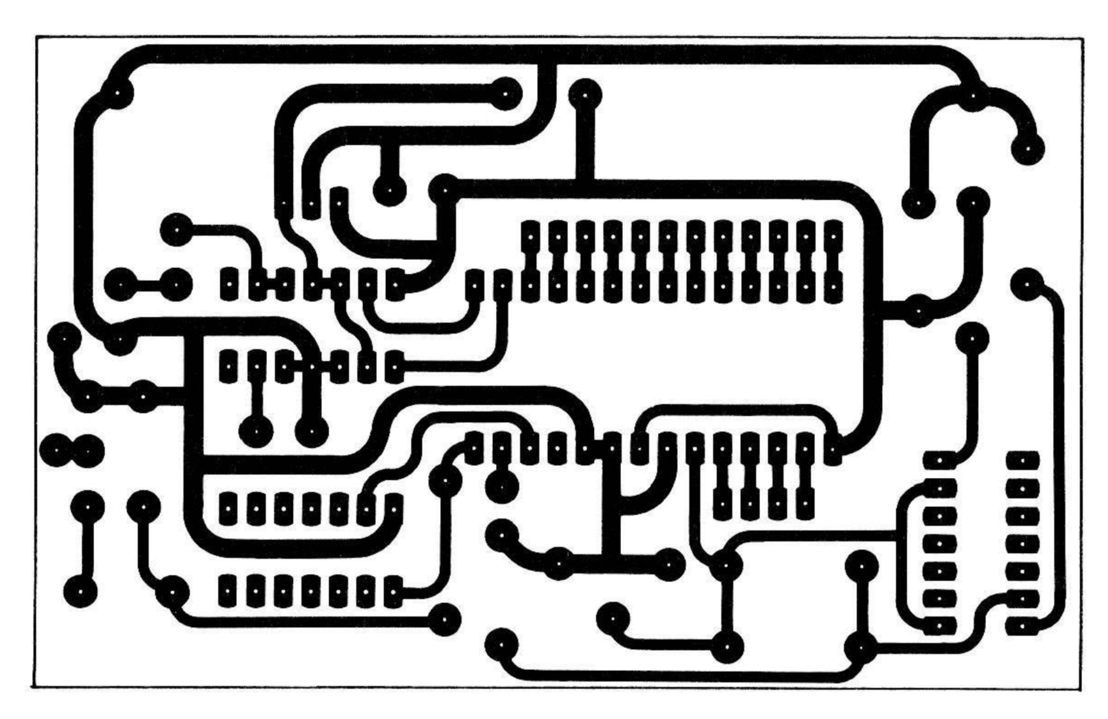


Fig. 3-6. — *Circuit imprimé.*

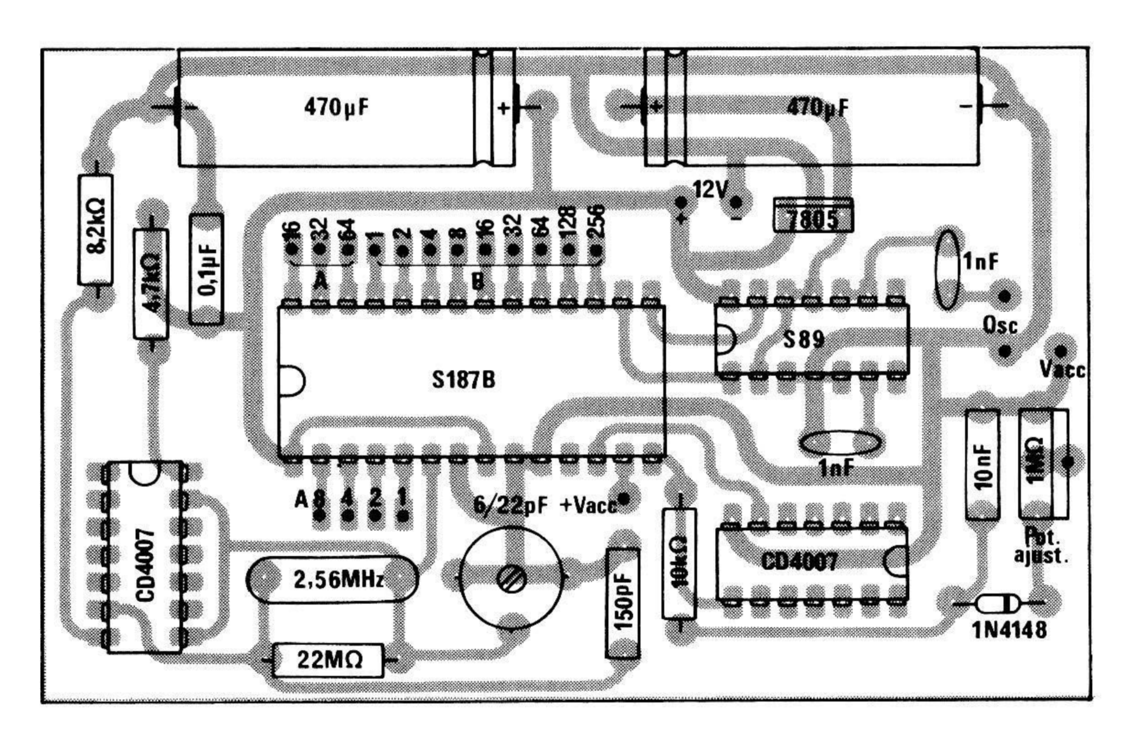


Fig. 3-7. — *Implantation.*

Avant toute chose, il faut bien sûr graver le circuit imprimé de la *figure 3-6*, et l'équiper conformément au plan de la *figure 3-7*.

Le choix de la fréquence du quartz est critique, car c'est à partir de sa valeur et d'un rapport de division interne au S 187 B qu'est obtenu le pas de 5 kHz.

Le tableau de la *figure 3-8* montre toutefois comment quelques autres valeurs peuvent être utilisées, au prix d'un câblage différent des broches 8, 9 et 10 du S 187 B. Pour ce qui est de l'interconnexion entre le synthétiseur et la carte d'interface, on se guidera sur le tableau de la *figure 3-9* qui donne également, pour information, le « poids » de chaque ligne dans le calcul de la fréquence.

Mise en œuvre

Le point le plus délicat reste bien sûr l'adaptation à l'oscillateur local du récepteur ou au VFO de l'émetteur. La *figure 3-10* donne un exemple de VFO pour la bande des 27 MHz, spécialement étudié pour être raccordé au synthétiseur. Dans bien des cas, cependant, il suffit de bobiner trois ou quatre spires de fil isolé par-dessus le bobinage oscillateur, pour disposer d'un signal très correct pour le S 89.

La broche 13 du S 187 B permet d'utiliser pour la commande de la varicap toute tension disponible mais *inférieure à 15 V*. Dans bien des cas, toutefois, le + 12 V général suffira, puisque le synthétiseur rattrapera toute dérive imputable à cette tension.

Une fois l'interconnexion établie et une fréquence d'essai programmée, il faut « accrocher » l'oscillateur sur le synthétiseur en agissant sur les éléments LC ajustables et sur le potentiomètre du synthétiseur. Dès lors, il n'y aura plus à se préoccuper de l'oscillateur, totalement placé sous la dépendance du système digital. Au moment de ces réglages, on ne saurait trop conseiller le contrôle au moyen d'un fréquencemètre précis. L'examen de la tension de commande de la varicap permet, pour sa part, de visualiser clairement le verrouillage de la boucle PLL.

br 10	br 9	br 8	rapport de division	f quartz exacte pour pas de 5 kHz
0	0	0	2048	10,240 MHz
0	1	0	1000	5,000 MHz
0	0	1	512	2,560 MHz
0	1	1	256	1,280 MHz
1	0	0	2560	12,800 MHz
1	1	0	1250	6,250 MHz
1	0	1	640	3,200 MHz
1	1	1	320	1,600 MHz

Fig. 3-8. — *Utilisation de quartz d'autres fréquences.*

BUS ZX 81	sorties carte interface	entrées carte synthétiseur	poids en fréquence
D_0	1	A_1	5 kHz
D_1	2	A_2	10 kHz
D_2	3	A_4	20 kHz
D_3	4	A_8	40 kHz
D_4	5	A_{16}	80 kHz
D_5	6	A_{32}	160 kHz
D_6	7	A_{64}	320 kHz
D_7	8	N.C.	—
A_0	9	B_1	500 kHz
A_1	10	B_2	1 MHz
A_2	11	B_4	2 MHz
A_3	12	B_8	4 MHz
A_4	13	B_{16}	8 MHz
A_5	14	B_{32}	16 MHz
A_6	15	B_{64}	32 MHz
A_7	16	B_{128}	64 MHz
A_8	17	B_{256}	128 MHz
A_9	18	N.C.	—
A_{10}	19	N.C.	—
A_{11}	20	N.C.	—
A_{12}, A_{13}, A_{14}, A_{15}	décodage 45056 (adresse de base)		

Fig. 3-9. — *Tableau d'interfaçage.*

Photo ci-contre
Vingt sorties « tout ou rien » permettent au ZX 81 de piloter des équipements passablement complexes, comme ce synthétiseur HF.

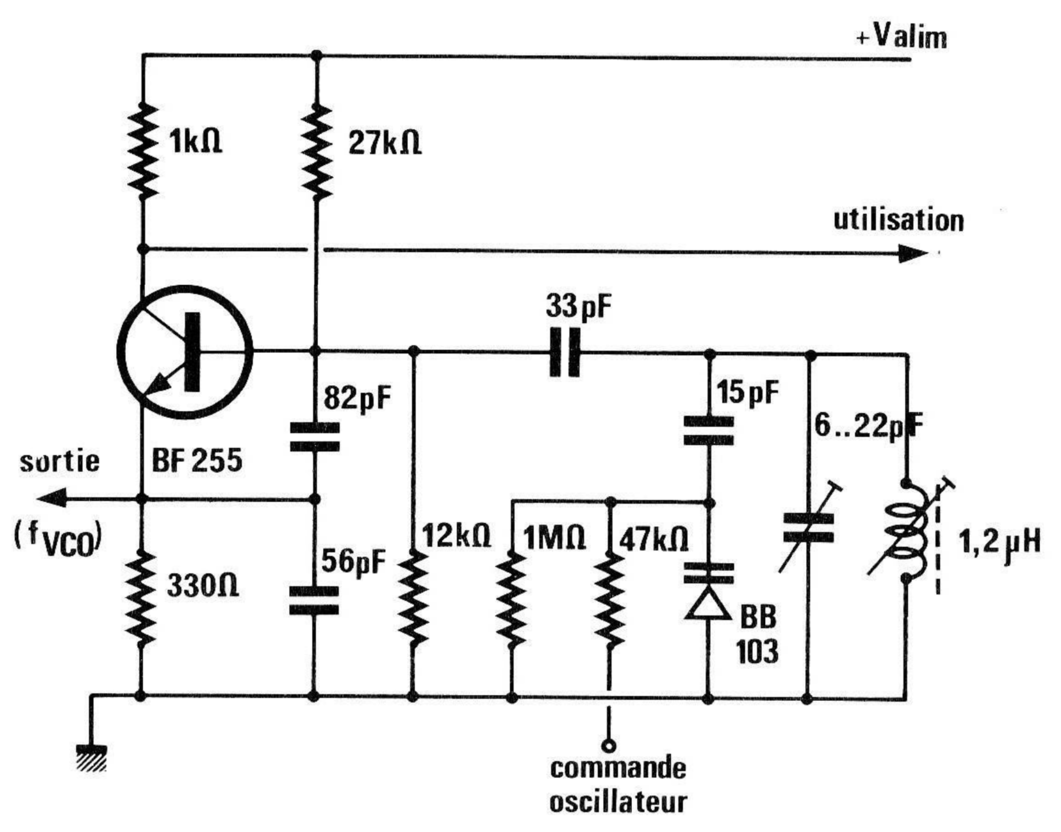


Fig. 3-10. — *Un exemple d'oscillateur 27 MHz.*

Quelques programmes

Sans programme d'accompagnement, tout ce système reste à peu près inutilisable. C'est le logiciel, plus ou moins sophistiqué, qui lui sera associé, qui lui confèrera toute la puissance dont il est capable :

Le programme de la *figure 3-11* se contente de provoquer la synthèse de la fréquence entrée au clavier, après un très simple dialogue. Notons que la machine informe l'utilisateur des éventuels « arrondis » auxquels elle se livre automatiquement si la fréquence demandée n'est pas multiple de 5 kHz.

```
   1 REM SYNTHETISEUR HF
  10 PRINT AT 10,4;"FREQUENCE A
SYNTHETISER ?"
  20 PRINT AT 13,8;"( EN MEGAHER
TZ )"
  30 INPUT F
  40 LET N=F*2
  50 LET B=INT N
  60 LET A=INT (.5+(100*(N-B)))
  70 POKE (45056+B),A
  80 CLS
  90 PRINT AT 10,5;"VOICI DU ";.
005*(A+100*B);" MHZ"
 100 PRINT AT 20,4;"POUR CHANGER
 NEWLINE"
 110 INPUT A$
 120 CLS
 130 GOTO 10
 140 REM COPYRIGHT 1983
```

Fig. 3-11.

Le programme de la *figure 3-12* présente une application beaucoup plus évoluée, le balayage permanent d'une bande de fréquences spécifiée.

Comme il s'agit là de réception, il est prévu de « déclarer » à la ligne 160 la valeur de la moyenne fréquence du superhétérodyne, afin que la fréquence s'affichant sur l'écran soit bien celle effectivement reçue.

Il est facile de modifier les paramètres que sont la vitesse de balayage ou le pas d'incrémentation de la fréquence.

Ce programme peut n'être considéré qu'en tant que routine devant être incorporée à un logiciel beaucoup plus perfectionné.

On pourrait imaginer, par exemple, de relier la sortie BF du récepteur à l'entrée cassette du ZX 81 qui, grâce à un sous-programme écrit en

```
  10 REM BALAYAGE
  20 PRINT "FREQUENCE BASSE EN M
HZ ?"
  30 INPUT FB
  40 CLS
  50 PRINT "FREQUENCE HAUTE EN M
HZ ?"
  60 INPUT FH
  70 CLS
  80 PRINT AT 10,6;"BALAYAGE ENT
RE"
  90 PRINT AT 13,0;FB;"        ET",F
H;" MHZ"
 100 PRINT AT 16,6;"DE 5 EN 5  M
HZ"
 110 FOR F=FB TO FH+.005 STEP .0
05
 120 LET N=F*2
 130 LET B=INT N
 140 LET A=INT (.5+(100*(N-B)))
 150 PRINT AT 0,0;F;" MHZ          "
 160 LET FI=-.455
 170 LET FR=F+FI
 180 LET N=FR*2
 190 LET B=INT N
 200 LET A=INT (.5+(100*(N-B)))
 210 POKE (45056+B),A
 220 FOR G=0 TO 100
 230 NEXT G
 240 NEXT F
 250 GOTO 110
 260 REM COPYRIGHT 1983
```

Fig. 3-12.

langage machine, viendrait tester la présence ou l'absence de trafic sur la fréquence reçue. Dès lors, cette fréquence pourrait être maintenue sur écoute, ou imprimée sur papier associée à l'heure, à la durée d'occupation, etc.

Mais il ne s'agit là que d'un exemple : un synthétiseur piloté par ordinateur ayant des possibilités encore bien plus étendues, il est vraisemblable que nos lecteurs ne seront guère embarrassés pour en tirer profit...

Pour les questions purement radioélectriques, on pourra se reporter à notre ouvrage *Réalisez vos récepteurs en circuits intégrés*, paru chez le même éditeur.

Une entrée par la prise cassette

En dehors de son clavier, le ZX 81 ne dispose normalement d'aucun moyen lui permettant de recevoir des « nouvelles » du monde extérieur. Il existe bien sûr des modules d'entrée-sortie adaptables, mais cette solution est souvent luxueuse par rapport aux besoins de l'application envisagée.

Nous allons décrire ici un procédé très simple permettant d'utiliser la prise EAR (magnétophone) du ZX 81 pour communiquer avec un programme en cours de déroulement.

1) A la découverte de la ROM

La mémoire morte (ROM) du ZX 81 contient le programme rédigé en langage machine, permettant à l'ordinateur d'avoir le comportement que nous lui connaissons. Avec une autre ROM, peut-être travaillerait-il en FORTH et non plus en BASIC ?

Il est extrêmement instructif (et tout aussi difficile !) de « désassembler » certaines des routines contenues dans cette partie de la mémoire, et sur le fonctionnement desquelles les fabricants du ZX gardent un mutisme aussi jaloux que total. On peut ainsi parvenir à savoir que, lorsqu'un signal BF de niveau suffisant est appliqué à la prise EAR, un octet 255 apparaît sur le port FE (254) du microprocesseur Z 80.

Ainsi donc, si nous arrivons à écrire un programme (en BASIC et langage machine) capable d'aller lire périodiquement ce port, nous pourrons faire recevoir au ZX 81 des « nouvelles de l'extérieur » autrement que par son clavier.

L'essentiel du problème consiste à ramener au BASIC le contenu du port 254. Le plus rapide est d'utiliser la variable USR, donc de transiter d'abord par le registre A, puis par les registres B et C. La *figure 3-13* donne le détail de ces opérations, sous la forme de la liste d'assemblage, en code décimal, de la routine machine utilisée.

```
16514   IN A,(254)    219 254
16516   LD C,A         79
16517   LD B,Ø          6   Ø
16519   RET           2Ø1
```

Fig. 3-13.

```
  10 REM ......
  20 POKE 16514,219
  30 POKE 16515,254
  40 POKE 16516,79
  50 POKE 16517,6
  60 POKE 16518,0
  70 POKE 16519,201
  80 IF USR 16514=255 THEN GOTO
100
  90 GOTO 80
 100 PRINT AT 10,6;"SIGNAL BF DE
TECTE"
 110 REM COPYRIGHT 1982
```

Fig. 3-14.

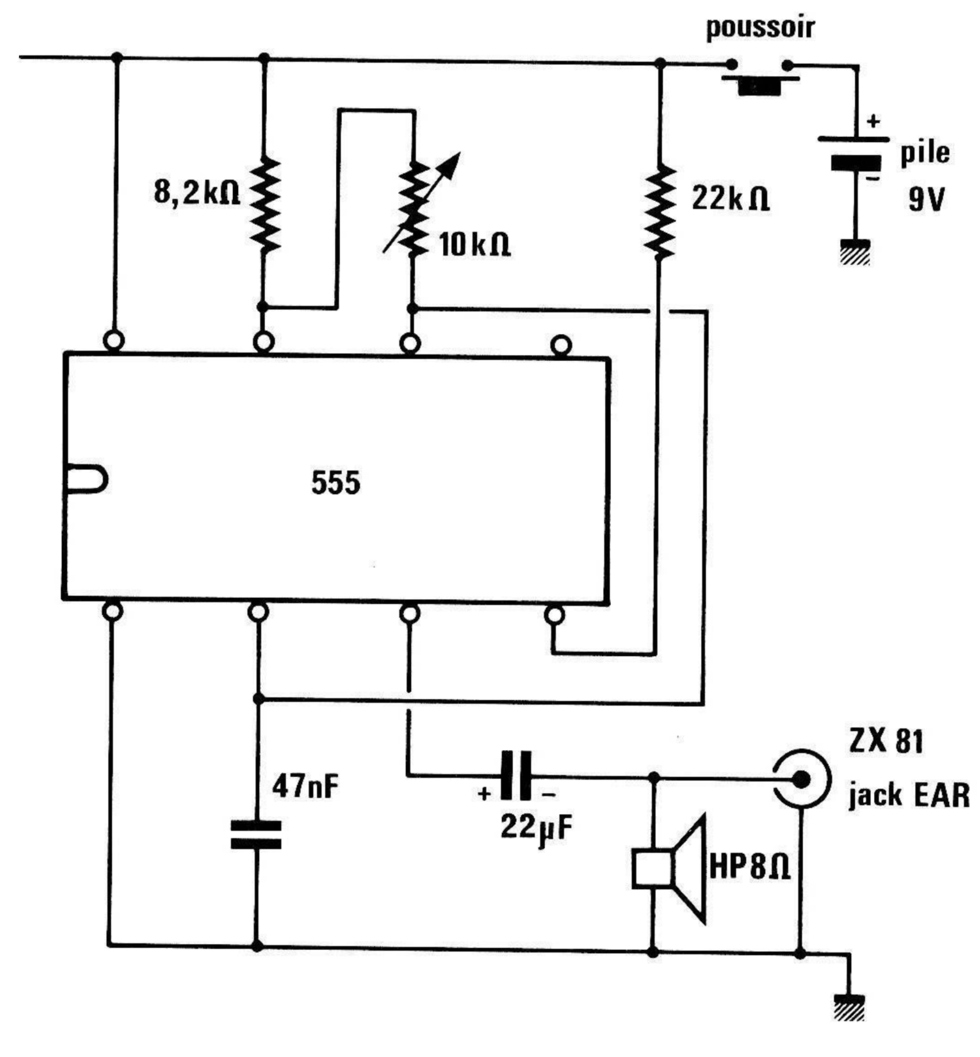


Fig. 3-15.

La *figure 3-14*, quant à elle, fournit un court programme BASIC, capable de charger cette routine dans une instruction REM, puis de la lancer de façon incessante, tant qu'un signal BF n'aura pas été identifié. A ce moment seulement, le programme exécutera la ligne 100 et s'arrêtera.

2) Réalisation d'une interface d'entrée

Détecter l'apparition d'un signal BF sur un jack est une chose, faire en sorte que ce signal apparaisse en est une autre !

Rares sont les cas où un tel signal est directement disponible. En pratique, il est plus commode de pouvoir détecter, par exemple, la fermeture d'un contact.

Le montage très simple, dont la *figure 3-15* donne le schéma de principe, est capable de fournir, par simple appui sur un bouton-poussoir, un signal BF parfaitement adapté à l'entrée EAR du ZX 81. Par ailleurs, la puissance disponible est suffisante pour permettre l'attaque simultanée d'un petit haut-parleur, ce qui peut être intéressant, notamment dans le domaine des jeux vidéo. C'est uniquement dans ce but qu'est prévu un réglage de fréquence, car le ZX peut accepter des signaux assez quelconques.

Ce petit circuit, qui fait usage d'un simple 555, peut être câblé sur un très petit circuit imprimé, dont le tracé des pistes est fourni par la *figure 3-16*. Le plan d'implantation de la *figure 3-17* ne nécessite pas de commentaire particulier.

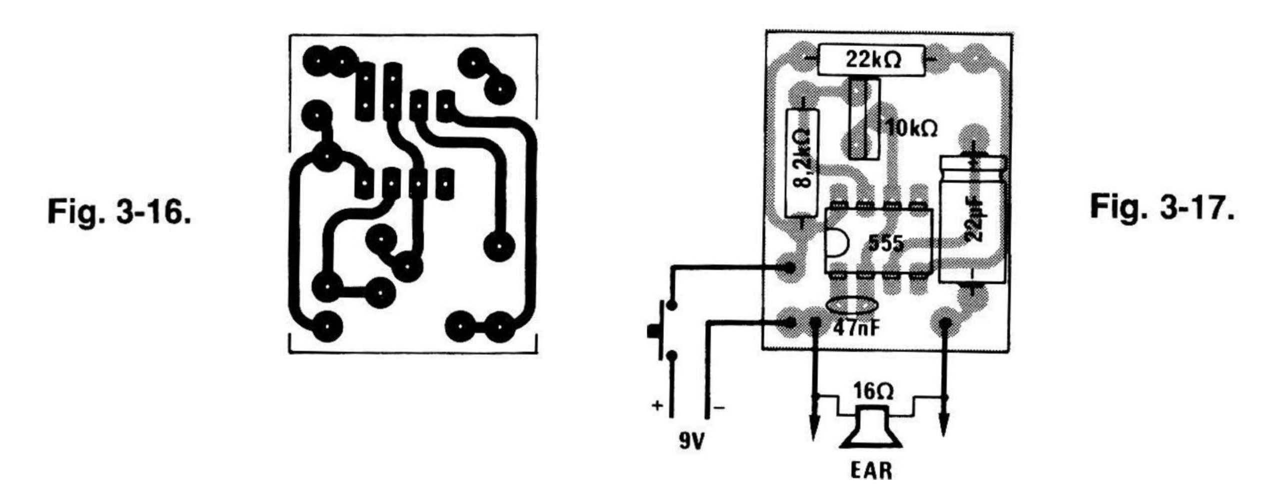


Fig. 3-16.

Fig. 3-17.

On pourra loger le montage, son haut-parleur, et sa pile, dans un très petit boîtier qui, nous allons le découvrir, pourra servir avantageusement de « poignée de jeu », puisque la longueur du cordon de raccordement est sans importance.

```
  1 REM JEU DE TIR
  2 POKE 16514,219
  3 POKE 16515,254
  4 POKE 16516,79
  5 POKE 16517,6
  6 POKE 16518,0
  7 POKE 16519,201
  8 LET P=0
  9 LET T=0
 10 LET L=INT (RND* 0)
 11 LET K=INT (RND*50)
 12 PLOT 10+K,L
 13 LET P=P+1
 15 FOR F=1 TO 40
 18 PLOT 0,F
 19 UNPLOT 0,F-1
 20 IF USR 16514=255 THEN GOTO
30
 21 IF USR 16514=255 THEN GOTO
30
 22 IF USR 16514=255 THEN GOTO
30
 23 IF USR 16514=255 THEN GOTO
30
 24 IF USR 16514=255 THEN GOTO
30
 25 NEXT F
 26 CLS
 29 GOTO 10
 30 FOR G=0 TO 60
 32 PLOT G,F
 35 IF G=10+K AND F=L THEN GOTO
 50
 40 NEXT G
 42 PRINT AT 0,0;"MANQUE"
 43 GOSUB 53
 44 CLS
 45 GOTO 10
 50 PRINT AT 0,0,"TOUCHE"
 51 LET T=T+1
 52 GOTO 43
 53 LET A$="S"
 55 PRINT
 56 PRINT AT 0,0;
 57 IF T=1 OR T=0 THEN LET A$="
"
 58 PRINT T;" TIR ;A$;" AU BUT
SUR ";P
 60 FOR F=1 TO 100
 65 NEXT F
 70 RETURN
 75 REM  COPYRIGHT 1982"
```

Fig. 3-18.

3) Un exemple d'application : un jeu de tir

Le programme de la *figure 3-18* fait apparaître à un endroit de l'écran, choisi de façon aléatoire par le ZX 81, un petit carré noir. En même temps se déplace verticalement un autre carré, matérialisant un tireur. Le jeu consiste à presser le poussoir de la poignée de jeu lorsque les deux points se trouvent sur une même horizontale. Attention, le « tir » intervient avec un très léger retard, et il faut donc un peu anticiper, mais pas trop ! On peut augmenter ce retard, donc la difficulté du jeu, en supprimant une ou plusieurs des lignes 21 à 24, ou le réduire, en ajoutant des lignes, quitte à effectuer une renumérotation du programme.

L'effet obtenu est d'autant plus intéressant que le haut-parleur de la poignée de jeu « sonorise » chaque tir !

Nos lecteurs trouveront sans aucun doute d'autres applications, ludiques ou « sérieuses », de ce procédé fort simple.

Chapitre 4

Réorganisons la mémoire



17 K-octets pour le prix de 16 !

Presque tous les utilisateurs de ZX 81 sont d'accord pour reconnaître qu'en dehors de la stricte initiation à la programmation, la mise en œuvre d'un module 16 K RAM (ou plus) s'impose pour la plupart des utilisations courantes.

Il est intéressant de remarquer que la mise en service d'une extension mémoire déconnecte automatiquement le boîtier RAM 1 K interne, par forçage à 1 de sa broche de sélection.

Dès lors, pourquoi ne pas tenter de lui redonner vie grâce à un petit montage très simple, qui introduit par la même occasion certains avantages plus que notables...

Organisation de la mémoire du ZX 81

Le ZX 81 est un ordinateur exceptionnellement économique, et ce résultat n'a pu être obtenu par ses créateurs que grâce à une totale maîtrise des coûts d'étude et de production.

S'il faut reconnaître qu'aucun compromis n'a été accepté sur le plan de la qualité, il est tout aussi certain que de nombreuses mesures simplificatrices ont été prises lors de la conception des circuits (voir chapitre 1).

En particulier, le schéma de la partie logique contenue dans le fameux « chip Sinclair » a été simplifié autant qu'il était humainement possible. Cela explique certaines « bizarreries » dans le comportement de la machine, qui passent totalement inaperçues lors d'une utilisation « sage », mais qui apparaissent lorsque l'on cherche à pousser le ZX dans ses derniers retranchements !

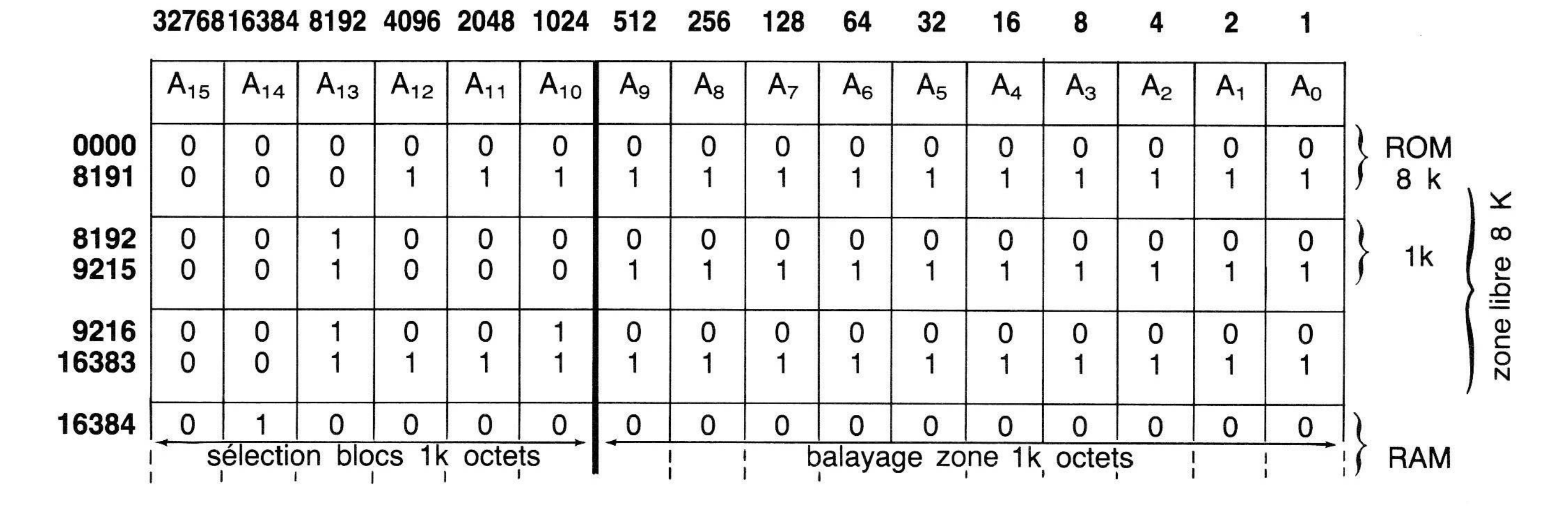
Par exemple, le manuel nous apprend que la mémoire morte (ROM) occupe les adresses mémoire 0 à 8 191, mais que la mémoire vive (RAM) se place entre 16 384 et 17 407 (ou 32 767 avec le module 16 K).

Ce « plan d'occupation » laisse dans l'ombre une zone de 8 K-octets comprise entre les adresses 8 192 et 16 383. Une exploration de ce mystérieux intervalle au moyen de commandes PEEK permet de constater que le contenu de la ROM apparaît une seconde fois entre ces deux limites.

La raison de ce phénomène tient en ce que le signal de sélection du boîtier ROM (nommé $\overline{ROMCS}$) est dérivé de celui présent sur la ligne d'adresse A14. Seulement, A14 reste à zéro jusqu'à l'adresse 16 383 alors que la ROM se termine à 8 191.

Les ingénieurs de chez Sinclair ont visiblement préféré perdre 8 K-octets d'espace mémoire, et bénéficier ainsi de l'économie de quelques portes dans leur circuit intégré spécifique, à moins qu'ils n'aient eu une autre idée en attente, telle qu'un BASIC « étendu » de 16 K ?

Quoi qu'il en soit, nous allons découvrir, avec l'aide de la *figure 4-1*, que cette zone abandonnée n'est pas forcément perdue pour tout le monde !



	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ROM 8 k
8191	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
8192	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1k (zone libre 8 K)
9215	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	
9216	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	
16383	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
16384	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RAM

Fig. 4-1.

Il est indispensable de construire de tels tableaux dès que l'on entreprend des interventions sur l'organisation de la mémoire d'un ordinateur.

Le microprocesseur Z 80 possède seize « lignes d'adresse » notées AØ à A15, et permettant donc de sélectionner jusqu'à 64 K adresses. Rien n'empêche de se représenter cette sélection de différentes façons, et notamment en imaginant que les lignes A10 à A15 servent à identifier un bloc de 1 024 octets parmi les 64 existants, alors que les lignes AØ à A9 servent à « balayer » les 1 024 octets de chaque bloc ainsi sélectionné.

Ainsi, la ROM 8 K pourrait être imaginée comme composée de huit blocs de 1 024 octets, identifiés par les combinaisons suivantes de A15, A14, A13, A12, A11 et A10 :

0 0 0 0 0 0
0 0 0 0 0 1
0 0 0 0 1 0
0 0 0 0 1 1
0 0 1 0 0 0
0 0 0 1 0 1
0 0 0 1 1 0
0 0 0 1 1 1

Pour sa part, la RAM 1 K serait identifiée uniquement par la combinaison : 010000, alors que le premier bloc de 1 024 octets de la zone inutilisée correspondrait à la combinaison :

001000

En réalité, le « chip Sinclair » se contente de sélectionner la ROM lorsque A14 est à zéro, et la RAM lorsque A14 est à un. Si nous décidions, par un câblage approprié, de modifier ce choix, nous pourrions « déplacer » le bloc de 1 024 octets constitué par le boîtier de RAM, n'importe où dans la zone adressable par le Z 80, puisque ce boîtier lui-même ne dispose que des lignes d'adresse AØ à A9 (voir *figures 1-5 et 1-6*), lesquelles, nous l'avons vu, ne servent qu'à son « balayage », sans aucun lien avec la place occupée par les 1 024 octets dans le « plan d'occupation mémoire ».

Eh bien, c'est exactement le genre d'exercice auquel nous allons nous livrer, avec, nous allons le découvrir, un certain nombre d'avantages.

La modification proposée

Il aurait été permis de supposer que le bloc 16 K prévu pour augmenter la capacité mémoire du ZX 81 n'était muni, en fait, que de 15 K-octets de ROM venant en complément du 1 K-octet d'origine.

En réalité, le bloc d'extension contient bien 16 K-octets, mais la mémoire disponible après son raccordement est toujours de 16 K-octets !

L'explication est simple : par la broche 2A de son connecteur, le module 16 K impose un niveau + 5 volts à la broche $\overline{\text{RAMCS}}$ du boîtier 1 K RAM, ce qui le met complètement hors service. A côté de cela, le bloc 16 K élabore son propre signal $\overline{\text{RAMCS}}$ à partir des lignes d'adresse du Z 80, dont il dispose en totalité. Il existe de nombreux avantages à remettre en service, moyennant certains artifices, le boîtier 1 K normalement « laissé pour compte »...

En premier lieu, il n'est pas désagréable de disposer de 17 K-octets pour le prix de 16 K !

Cependant, l'intérêt essentiel de la manipulation est qu'il est possible de « déplacer » ce bloc de 1 024 octets « récupérés », pour le placer dans la fameuse zone inutilisée de 8 K-octets. En effet, cette zone est entièrement ignorée par le BASIC, et seuls des POKE et des PEEK permettront de communiquer avec elle, un peu à la manière des espaces mémoire placés au-dessus de RAMTOP. Comme ces derniers, ce nouveau bloc de RAM ne pourra pas être sauvegardé sur cassette, mais son contenu pourra toujours être transféré, pour les besoins de la cause, dans une zone accessible à la routine SAVE.

En revanche, le point le plus intéressant est que notre bloc « ajouté » est totalement à l'abri des procédures d'initialisation du ZX 81.

On sait que le microprocesseur Z 80 commence toujours son premier programme à l'adresse Ø, notamment lors de la mise sous tension du ZX 81. On peut appeler ce programme dit « d'initialisation » en frappant, n'importe quand, la commande suivante :

RAND USR Ø newline

De cette façon, on aura tôt fait de se rendre compte que cela efface tout ce qui pourrait se trouver en mémoire, même au-dessus de RAMTOP.

Par contre, notre zone de 1 K-octet « protégée » survit à toute tentative d'effacement, que ce soit par le programme d'initialisation, par NEW, ou par le chargement d'un programme ! Seule une coupure d'alimentation pourra en venir à bout.

Nous disposons donc là d'un espace exceptionnellement sûr, pour ranger des données importantes, où surtout des PROGRAMMES EN LANGAGE MACHINE, qui nous seront très bientôt fort utiles...

En effet, quiconque s'est livré aux joies de la programmation en assembleur sur le ZX 81 sait forcément que le lancement d'un tel programme avant sa mise au point complète débouche neuf fois sur dix

sur un blocage de la machine obligeant à couper le courant. Une solution consiste à monter un poussoir de remise à zéro de l'unité centrale (en parallèle sur C5), mais son utilisation efface toute la mémoire !

La zone protégée dont nous vous proposons la création est tout à fait à l'abri des remises à zéro de l'unité centrale. Aussi, en cas de blocage de la machine, on peut revenir au curseur en utilisant ce poussoir de RAZ, puis lister, modifier, et relancer à loisir le programme en cours de mise au point. Une fois cette mise au point terminée, on pourra « figer » ce logiciel dans une mémoire EPROM, grâce au programmateur que nous décrirons.

La *figure 4-1* montre que pour obtenir les résultats escomptés, il suffit d'empêcher la sélection de la ROM lorsque A13 est à un, afin de libérer la zone inutilisée de 8 K-octets, et de valider le boîtier 1 K RAM lorsque la combinaison suivante sera réalisée :
— module 16 K en place,
— A13 = 1
— A10, A11, A12, A14 = Ø,
— MREQ = Ø

Le signal $\overline{\text{MREQ}}$ indique que l'unité centrale a besoin d'échanger des informations avec la mémoire, et rentre toujours dans la composition de signaux tels que $\overline{\text{RAMCS}}$ ou $\overline{\text{ROMCS}}$ (voir figure 1-2). On notera que A15 n'est pas utilisée, car ce n'est qu'avec des blocs d'extension de capacité supérieure à 16 K-octets que cette ligne pourrait passer à un niveau haut, et de tels blocs possèdent souvent des zones protégées, rendant notre modification inutile. On remarquera aussi qu'en l'absence de bloc 16 K, la RAM interne se bloque, empêchant la machine de fonctionner, afin de signaler cet oubli à l'opérateur.

Réalisation pratique

Les fonctions logiques à réaliser étant très simples, le montage n'utilise que deux boîtiers TTL très courants, et quelques composants annexes. Le schéma de la *figure 4-2* ne fait que concrétiser les conditions logiques précédemment énoncées.

La mise en œuvre pratique de la modification devra se dérouler en deux étapes :

D'une part, la construction d'un petit module sur un circuit imprimé gravé d'après la *figure 4-3*, et câblé conformément au plan de la *figure 4-4*.

D'autre part, la modification du ZX 81 en vue de l'incorporation de ce module dans son boîtier (il reste suffisamment de place libre, notamment sous le clavier).

+ 5 V

BC 107

$\overline{MREQ}$

A15

A14

A13

A12

A11

$\overline{ROMCS}$

S

(vers $\overline{RAMCS}$)

74 LS 30

5/6 74 LS 04

Fig. 4-2. — *N.B. Pour la récupération de 1 k RAM sur le ZX 81, couper la piste du connecteur reliant la broche 2A de la machine au bloc d'extension mémoire.*

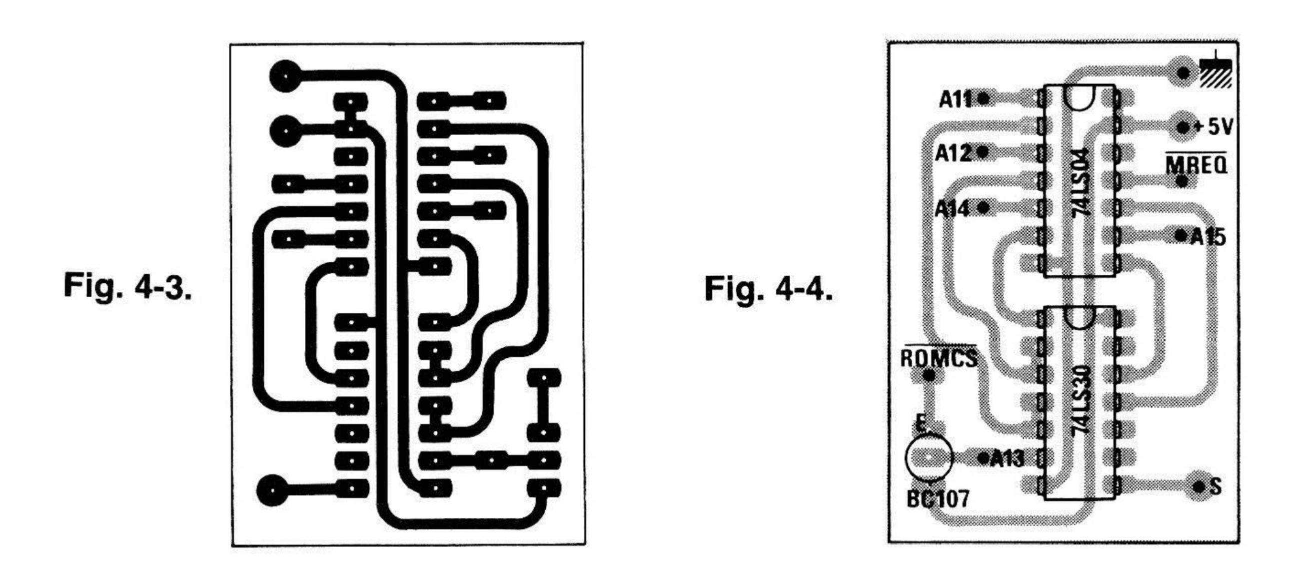


Fig. 4-3.

Fig. 4-4.

Il est nécessaire de couper, avec un outil tranchant mais précis, deux pistes du circuit imprimé d'origine, situées chacune sur une face de la carte. Il s'agit de la piste reliant la résistance R 2 à la broche 20 du boîtier RAM (ou aux broches 8 des deux boîtiers RAM, selon la version dont on dispose), et de celle reliant ces mêmes broches 20 ou 8 au contact 2 A du connecteur arrière.

Cela terminé, il pourra être intéressant de souder sur la carte de petits plots à souder ou à wrapper, aux points suivants, faciles à repérer au moyen du schéma du ZX 81 et de la *figure 4-2* :

— $\overline{ROMCS}$ après R 28,
— $\overline{RAMCS}$ au niveau du ou des boîtiers de RAM,
— broche 2 A du connecteur,
— A10, A11, A12, A13, A14, $\overline{MREQ}$,
— + 5 V et masse.

Il ne reste plus, alors, qu'à raccorder le module à ces différents points et à passer aux essais, non sans avoir monté le très utile poussoir de RAZ du Z 80 en parallèle avec C 5.

Mise en service

Cette adaptation effectuée, le ZX 81 doit fonctionner (avec son module 16 K), comme si rien n'avait été fait.

Les choses changent, cependant, si l'on entre le petit programme de la *figure 4-5*.

```
10 REM ESSAI ZONE PROTEGEE
20 POKE 8192,195
30 POKE 8193,0
40 POKE 8194,32
50 INPUT A$
60 RAND USR 8192
70 REM COPYRIGHT 1982
```

Fig. 4-5.

Lancé sur un ZX 81 non modifié, ce programme échoue à ses trois tentatives de POKE et déclenche, à la ligne 60, l'effacement complet de la mémoire puisque l'adresse 8 192 est une sorte de « fantôme » de l'adresse Ø.

Sur un ZX 81 modifié comme il a été expliqué, les lignes 20, 30 et 40 du programme chargent dans la zone protégée le court programme machine suivant :

8 192 JP 8 192,

qui est aussitôt lancé par la ligne 60, après appui sur NEWLINE. Or, il s'agit d'un programme « en boucle », incapable de s'arrêter seul. Habituellement, un tel programme bloque la machine jusqu'à ce qu'une main secourable vienne couper l'alimentation !

Avec notre modification, on peut utiliser le poussoir de RAZ, ce qui efface bien sûr la partie BASIC du programme, mais garde intacte la routine machine.

Pour s'en convaincre, il suffit de faire manuellement :

RAND USR 8 192 newline,

ou encore :

PRINT PEEK 8 192 newline
PRINT PEEK 8 193 newline
PRINT PEEK 8 194 newline

Grâce à cette toute petite démonstration, nos lecteurs amateurs de programmation en assembleur auront pu constater l'intérêt que revêt pour eux l'utilisation de la nouvelle zone mémoire dont ils disposent à présent, au prix d'une adaptation vraiment mineure de leur ZX 81.

Application pratique : un générateur d'instructions REM

En plus des avantages qui viennent d'être décrits, le fait de disposer de cette nouvelle zone de mémoire permet de mener à bien des tâches tout à fait particulières. Notamment, un programme situé dans cette partie de la mémoire peut fort bien intervenir profondément dans le contenu de la RAM « normale », alors qu'il se perturberait lui-même s'il y résidait !

Le programme en assembleur listé sur la *figure 4-6* est capable de créer des instructions REM contenant 250 points, et ce en quelques secondes. Quiconque aura déjà effectué ce travail à la main saisira immédiatement l'intérêt de la chose...

Le programme BASIC de la *figure 4-7* charge la routine machine dans la zone protégée, puis s'efface lui-même pour ramener la machine au curseur. Il suffit alors d'entrer les lignes 10 et 20 du court programme de la *figure 4-8*, et de lancer un RUN pour que la machine s'arrête au bout de quelques instants sur un étrange compte rendu C/6939, prouvant bien que le moniteur s'est quelque peu affolé.

Une simple pression sur NEWLINE fait tout rentrer dans l'ordre, et édite une image semblable à celle de la *figure 4-9*.

Si 250 octets ne suffisent pas à l'opérateur, rien n'est plus simple que de dupliquer la ligne 10 autant de fois que nécessaire au moyen de la fonction EDIT, ainsi qu'en témoigne la *figure 4-10*.

```
*D

=2000
2000 LDA  407F
2003 INC  A
2004 CP   C8;
2006 RET  Z
2007 STA  407F
200A LD   A,1B; .
200C LD   HL,4083
200F CALL 0526
2012 JP   2000
2015 NOP
2016 NOP
2017 NOP
Q
*G
```

Fig. 4-6.

```
   10 FAST
   20 LET A$="058127064060254200300050127064062027033131064205038005195000032"
   30 FOR F=1 TO LEN A$/3
   40 POKE 8191+F,VAL A$((3*F)-2 TO 3*F)
   50 NEXT F
   60 NEW
   70 REM COPYRIGHT 1983
```

Fig. 4-7.

```
10 REM ...
20 RAND USR 8192
30 REM COPYRIGHT 1983
```

Fig. 4-8.

```
   10 REM .......................
................................
................................
................................
................................
................................
...............
   20 RAND USR 8192
   30 REM COPYRIGHT 1983
```

Fig. 4-9.

Une fois l'espace nécessaire réservé, il est alors possible de transférer, grâce aux fonctions PEEK et POKE, des routines machine préalablement mises au point dans la zone protégée, dans les lignes REM ainsi créées. Il deviendra ainsi facile de les lister, et surtout de les sauvegarder sur cassette. Bien sûr, cette pratique exige un emploi massif de l'adressage relatif puisque les adresses absolues ne seront plus les mêmes après le transfert.

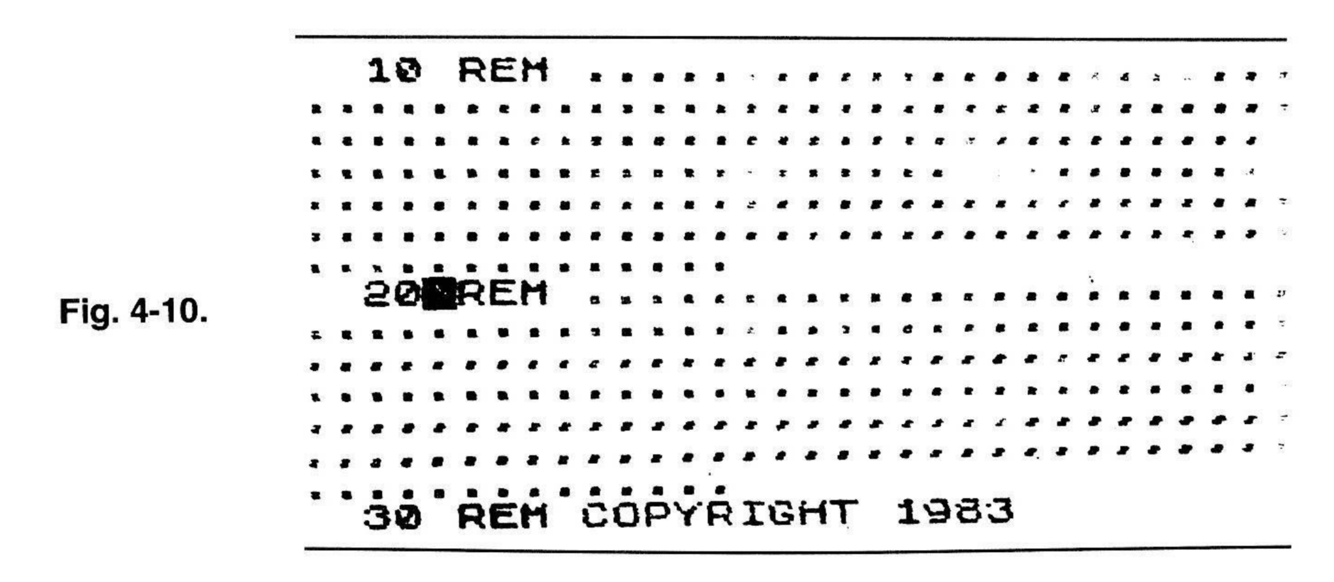


Fig. 4-10.

Annexe : adaptation à l'extérieur du ZX 81

Ceux de nos lecteurs qui ne souhaiteraient pas intervenir sur le circuit imprimé de la machine pourront raccorder ce petit adaptateur sur un simple connecteur « gigogne » du modèle de ceux qui seront utilisés massivement dans la suite de ce livre. Il n'est alors pas nécessaire d'interrompre la liaison entre RAMCS et le boîtier de RAM incorporé, mais il est par contre *indispensable* de couper la liaison reliant les deux broches référencées 2 A du connecteur. La sortie S de l'adaptateur sera reliée à la broche 2 A du connecteur femelle (côté machine), alors que son homologue côté extension 16 K restera « en l'air ».

Ce montage purement externe présente l'inconvénient d'ajouter une « couche » d'extensions essentiellement branlantes, mais facilite beaucoup le retour à la configuration d'origine, lors de manipulations devant se dérouler sur un ZX 81 muni de sa seule mémoire 1 K RAM d'origine.

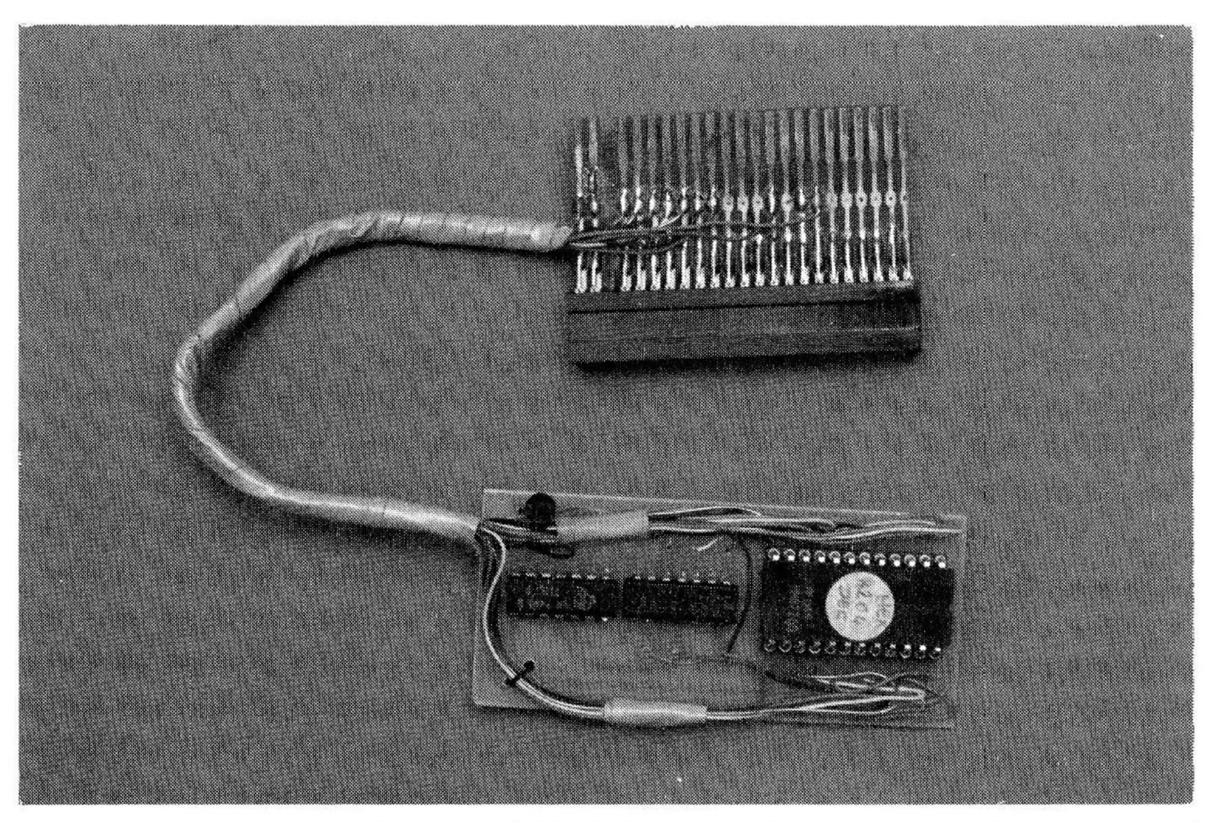
Au lecteur de choisir, donc !

Adaptation d'une EPROM 2716 au ZX 81

Tout ordinateur individuel possède une mémoire plus ou moins vaste, mais invariablement séparée en deux parties, la ROM (ou mémoire morte) et la RAM (ou mémoire vive).

Traditionnellement, la ROM abrite le programme de base nécessaire dans tous les cas au fonctionnement de la machine (moniteur, interpréteur, etc.).

Les programmes de l'utilisateur sont généralement chargés en RAM avant d'être lancés, soit par frappe au clavier, soit à partir du lecteur de cassettes. Dans les deux cas, et spécialement avec le ZX 81, il faut un temps non négligeable pour mener à bien ces opérations.



Cet adaptateur de mémoires EPROM ouvre la porte au changement de programmes « en cartouches » sur le ZX 81.

Certains ordinateurs peuvent recevoir des « cartouches » qu'il suffit d'embrocher pour que de nouveaux programmes soient immédiatement disponibles. Ces chargeurs ne sont autres que des ROM supplémentaires préprogrammées.

Avec le montage qui va être décrit, tout ZX 81 pourra ainsi accepter des ROM extérieures de 2 K-octets que l'utilisateur pourra programmer, voire effacer, comme il l'entendra, au même titre qu'une cassette.

1) ROM contre cassette

Si l'on excepte l'entrée au clavier, qui ne peut en fait convenir qu'à la toute première mise en mémoire, et la lecture de disquettes, réservée aux systèmes déjà évolués, il ne subsiste guère que deux procédés courants permettant de charger des programmes sur un ordinateur individuel :

La lecture de cassettes magnétiques est le procédé le plus répandu, car il ne réclame qu'un matériel très bon marché tout en restant très simple. L'approvisionnement en supports d'information (simples cassettes audio) ne pose aucun problème.

Par contre, le chargement des programmes reste désespérément lent (30 secondes par K-octet pour le ZX 81), et en cas de coupure d'alimentation ou de blocage de l'unité centrale, le programme est intégralement perdu, puisqu'il réside en RAM.

L'enfichage de modules préprogrammés (ROMs additionnelles) est massivement utilisé sur les ordinateurs de jeux, les traductrices de poche, et certaines calculatrices programmables. Les avantages du procédé sont déterminants : aucun appareil supplémentaire n'est nécessaire, chargement instantané et immunité totale contre tous incidents tels que perte du contenu de la RAM.

Au chapitre des inconvénients, on peut citer un coût un peu supérieur (encore qu'en choisissant bien son revendeur, on puisse acquérir des EPROMS de 2 K-octets pour le prix d'une très bonne cassette audio), et surtout la nécessité d'un équipement spécial pour la programmation et l'effacement.

Achetés dans le commerce, ces appareils reviennent relativement cher, car ils sont surtout étudiés en vue de besoins industriels.

Aussi, allons-nous décrire dans le prochain chapitre la réalisation pratique d'un programmateur exploitant au maximum les possibilités du ZX 81, grâce à l'utilisation d'un circuit dérivé de notre carte à vingt sorties étudiée au chapitre précédent.

Egalement, nous découvrirons que la construction d'un effaceur se ramène surtout à des opérations de menuiserie puisque la plupart des électriciens peuvent fournir les tubes ultraviolets (germicides) nécessaires, ainsi que leurs accessoires.

2) Adaptation d'une EPROM sur le ZX 81

Les EPROMS sont des ROM effaçables aux ultraviolets et reprogrammables aussi fréquemment qu'on le souhaite. Leur avantage majeur en ce qui nous concerne est qu'elles n'exigent, en lecture, qu'une seule tension d'alimentation de + 5 V, déjà disponible dans l'ordinateur. Dès lors, le principal rôle du circuit d'adaptation sera de permettre l'adressage de l'EPROM dans une zone convenable de la mémoire.

Dans le cas du ZX 81, il est commode de remarquer que le plan d'occupation de l'espace mémoire disponible réserve 16 K-octets à la ROM. Or, le BASIC Sinclair n'occupant que 8 K-octets, il est tout indiqué de loger notre ROM personnelle à la suite !

Notre circuit devra alors adresser l'EPROM pour les 2 048 octets suivant l'adresse 8 191, mais en même temps bloquer la ROM Sinclair qu'une ambiguïté de décodage adresse normalement deux fois !

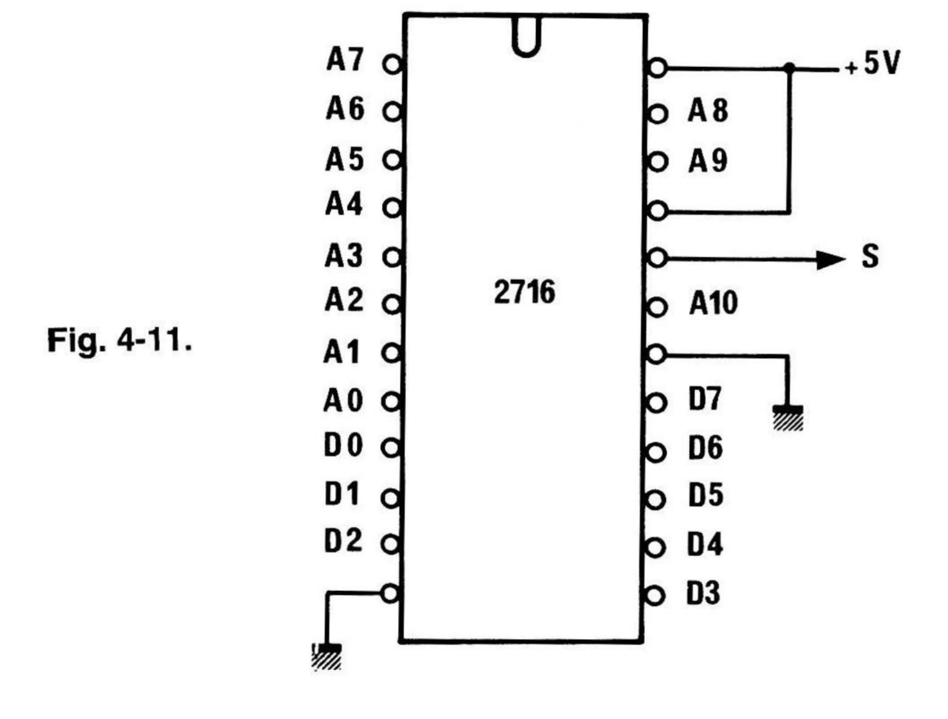


Fig. 4-11.

Or, cette tâche rappelle à s'y méprendre celle du montage précédent, à ceci près que sa sortie doit attaquer la broche de sélection de l'EPROM au lieu de la ligne $\overline{\text{RAMCS}}$ du ZX 81.

Le schéma de la *figure 4-11* récapitule les connexions à établir avec la mémoire EPROM de type 2 716, choisie en raison de son prix très abordable et de son excellente disponibilité auprès de la plupart des bons revendeurs de composants.

Le décodage effectué sur les lignes d'adresse A11 à A15 est complet, ce qui permet de conserver intact tout l'espace réservé à la RAM et ce, quelle que soit la capacité des éventuelles RAM additionnelles que l'on pourrait raccorder en même temps. Par contre, il ne faudrait pas utiliser simultanément des accessoires adressés entre 8 192 et 10 239.

La *figure 4-12* donne le tracé d'un circuit imprimé prévu pour recevoir l'EPROM et ses circuits associés sous une forme compacte, selon le plan de la *figure 4-13*. On pourra équiper ce module d'un connecteur gigogne (qui sera décrit dans un autre chapitre) ou bien le loger à demeure dans le boîtier de l'ordinateur, par exemple sous le clavier, si cette place très accueillante n'est pas déjà occupée !

En utilisant un support de bonne qualité, on pourra faire alterner plusieurs EPROMS sur le même adaptateur mais, compte tenu du faible coût de ce circuit, il serait plus confortable d'équiper chaque EPROM de son propre décodeur, afin de disposer de véritables « cartouches » enfichables.

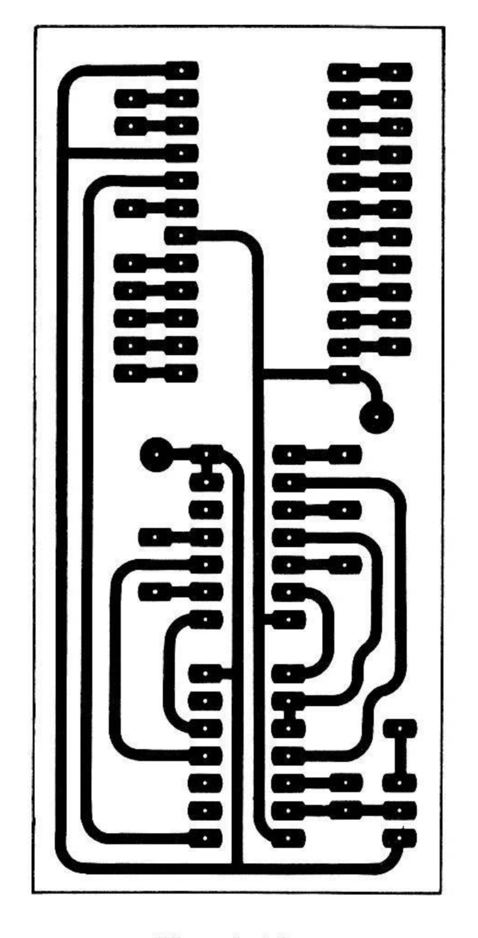


Fig. 4-12.

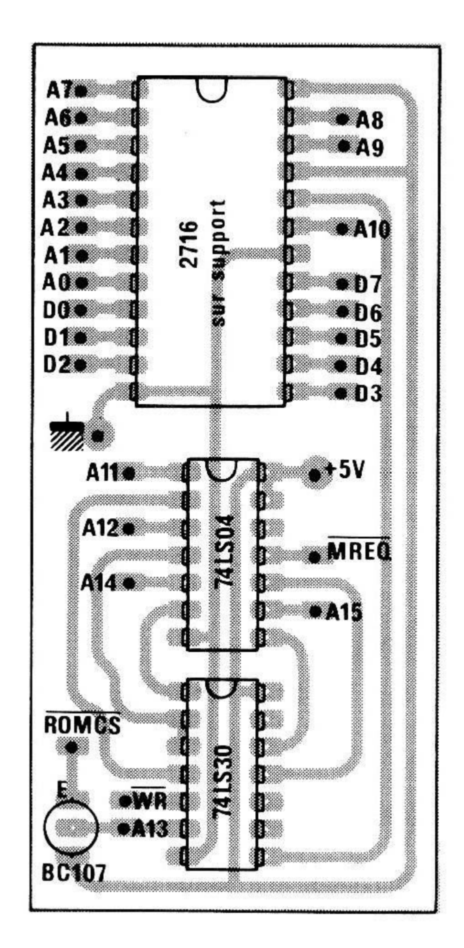


Fig. 4-13.

Que mettre dans l'EPROM ?

Compte tenu de la conception générale du ZX 81, le plus commode est de loger des programmes écrits en langage machine dans cette mémoire morte additionnelle.

Il est devenu banal, en Grande-Bretagne, d'ajouter ces « compléments » au BASIC d'origine, sous la forme de petites routines que l'on appelle par des RAND USR.

En ce qui nous concerne, un chapitre entier sera consacré au développement d'idées de logiciels machine, susceptibles de « résider » en EPROM, mais étroitement liés à l'exploitation des circuits d'entrée-sortie dans le cadre de toutes sortes d'automatismes utilitaires.

Sitôt la machine sous tension, un simple RAND USR 8 192 manuel suffira à lancer le programme, sans autre forme de chargement. Cependant, nous décrirons aussi, ce qui est beaucoup moins classique, une méthode permettant de « brûler » en EPROM de petits programmes BASIC dont l'écriture est bien évidemment à la portée de tous nos lecteurs.

Le même RAND USR suivi d'un RUN permettra l'appel immédiat du logiciel BASIC désormais « résident », sans le moindre recours à un lecteur de cassettes.

Remplacement de la ROM Sinclair par une EPROM

Cette étude pourrait s'intituler « que faire avec un ZX 81 qui aurait perdu sa ROM ? ». On sait que c'est la ROM Sinclair de 8 K-octets qui détermine entièrement le fonctionnement de la machine dès sa mise sous tension et jusqu'à son arrêt définitif. Enlevons-la, et l'ordinateur devient un banal circuit imprimé complètement inerte. Remplaçons-la par une autre, et c'est une toute nouvelle machine que nous allons découvrir !

Un ZX 81 (pas encore modifié !) et un programmateur très facile à construire sont les seuls outils nécessaires pour « fabriquer » par soi-même de nouvelles ROMS pour le petit Sinclair...

Quelques rappels sur la ROM Sinclair

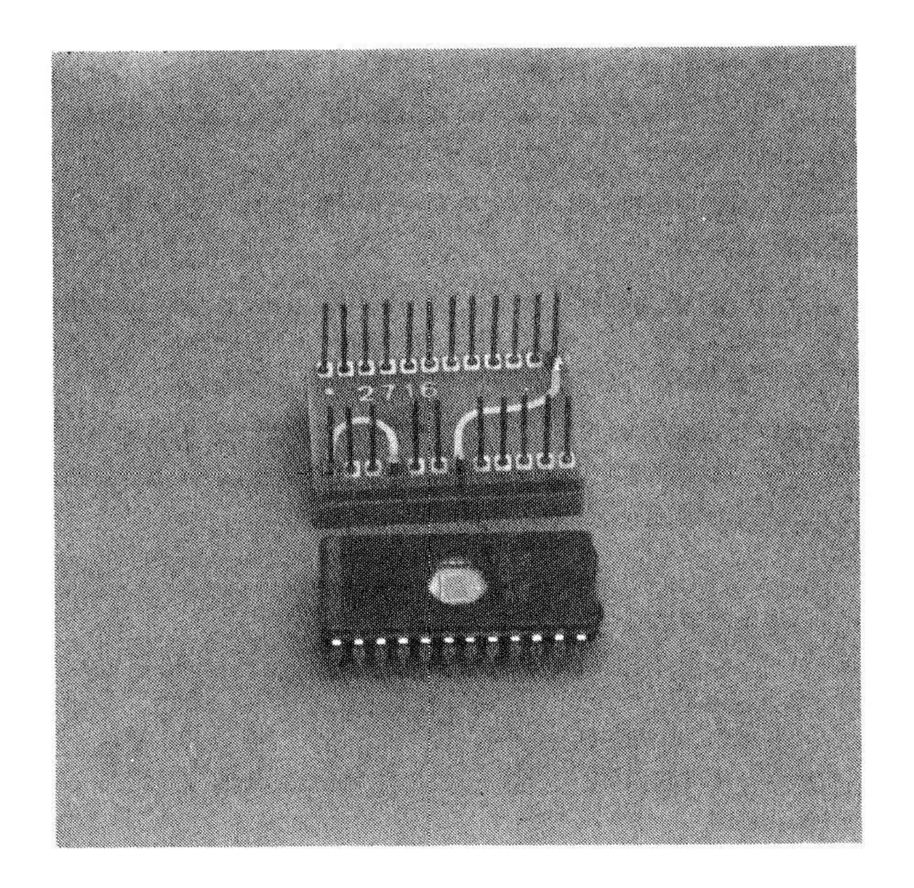
La mémoire inaltérable (ROM) du ZX 81 est implantée en mémoire à partir de l'adresse zéro. C'est dire qu'un microprocesseur Z 80 normalement constitué commence dès sa mise sous tension à exécuter le programme (en langage machine) qu'elle contient (8 K-octets allant jusqu'à l'adresse décimale 8 191).

En l'occurrence, ce programme « moniteur » commence par initialiser le système, puis attaque l'interpréteur BASIC permettant à l'utilisateur d'employer ce langage très simple.

Il suffirait (en théorie !) de monter en remplacement une ROM programmée comme il convient, pour que le ZX 81 travaille dans un langage tout différent.

Précisément, la firme COMPROCSYS commercialise une EPROM de type 2 532 (4 K-octets), contenant un logiciel extrêmement performant transformant le ZX 81 en un système de développement de Z 80. La mise au point de tels programmes ne concerne bien sûr que des informaticiens de haut vol (pas nécessairement professionnels d'ailleurs !), mais bien des applications plus simples peuvent utiliser l'idée du remplacement de la ROM.

L'avantage essentiel de cette solution est que l'exécution du programme commence automatiquement lors de la mise sous tension de la machine. Mieux, en cas de coupure d'alimentation, tout peut repartir sans intervention d'un quelconque opérateur !



Ce petit adaptateur permet de monter une mémoire 2716 programmée par l'utilisateur, en remplacement de la ROM Sinclair : nous allons découvrir une toute autre machine !

Evidemment, les possibilités du BASIC ne sont plus disponibles : l'affichage sur l'écran, par exemple, oblige à étudier des routines passablement complexes. Cependant, rien n'empêche de s'inspirer de celles contenues dans la ROM d'origine...

Remplacement de la ROM Sinclair par une 2716

Pourquoi utiliser une 2716 de préférence à d'autres types de mémoires mortes ?

Ce choix résulte de diverses remarques dont voici les principales :

- 2 K-octets suffisent en général plus que largement pour des applications courantes. Il s'agit bien, en effet, de langage machine, dont la compacité est proverbiale.
- Cette EPROM est électriquement compatible avec le ZX 81 : en lecture, elle se contente d'une alimentation 5 V unique.
- La possibilité d'effacer aux UV une mémoire dont le contenu a cessé de plaire est un élément à ne pas négliger.
- Enfin, on trouve des 2716 à peu près partout à des prix accessibles à tous, ce qui n'est pas forcément vrai pour les autres types qui auraient pu être retenus.

Bien sûr, on ne peut enficher directement une 2716 à la place de la ROM d'origine, mais il est intéressant de noter que les différences de brochage sont insignifiantes (norme BYTEWIDE).

Il est donc très facile de construire un adaptateur « piggy-back » qui viendra s'intercaler entre les deux.

Il s'agit d'un simple support à wrapper à 24 broches, dont certaines seront coupées, les autres amincies, et sur lequel on montera un très petit circuit imprimé chargé d'établir les liaisons nécessaires.

On commencera donc par graver le circuit imprimé dont le tracé est donné par la *figure 4-14*. Après l'avoir enfilé *dans le bon sens* et soudé au support, on l'amputera des broches 18 et 21.

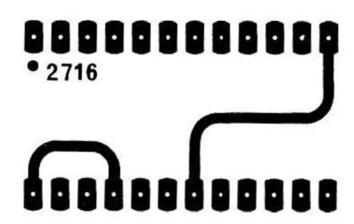


Fig. 4-14.

Cela fait, il faudra encore amincir l'extrémité des broches restantes de façon à faciliter leur insertion dans le support de la ROM du ZX 81. Cette opération est facile à réaliser au moyen d'un petit disque à meuler monté sur une perceuse à piles. A défaut, la lime est une solution tout à fait acceptable, quoique moins expéditive !

Il n'est pas indispensable de raccourcir les broches du support, car il ne sera de toute façon pas possible de refermer le boîtier de l'ordinateur : il pourra être commode de découper dans la coquille supérieure de son boîtier, une ouverture rectangulaire de laquelle le support dépassera tout juste, facilitant ainsi l'échange des 2716. Pour notre part, nous avons carrément coupé en deux cette coquille juste au-dessus du clavier, ce qui donne au circuit imprimé une accessibilité jamais vue... (1)

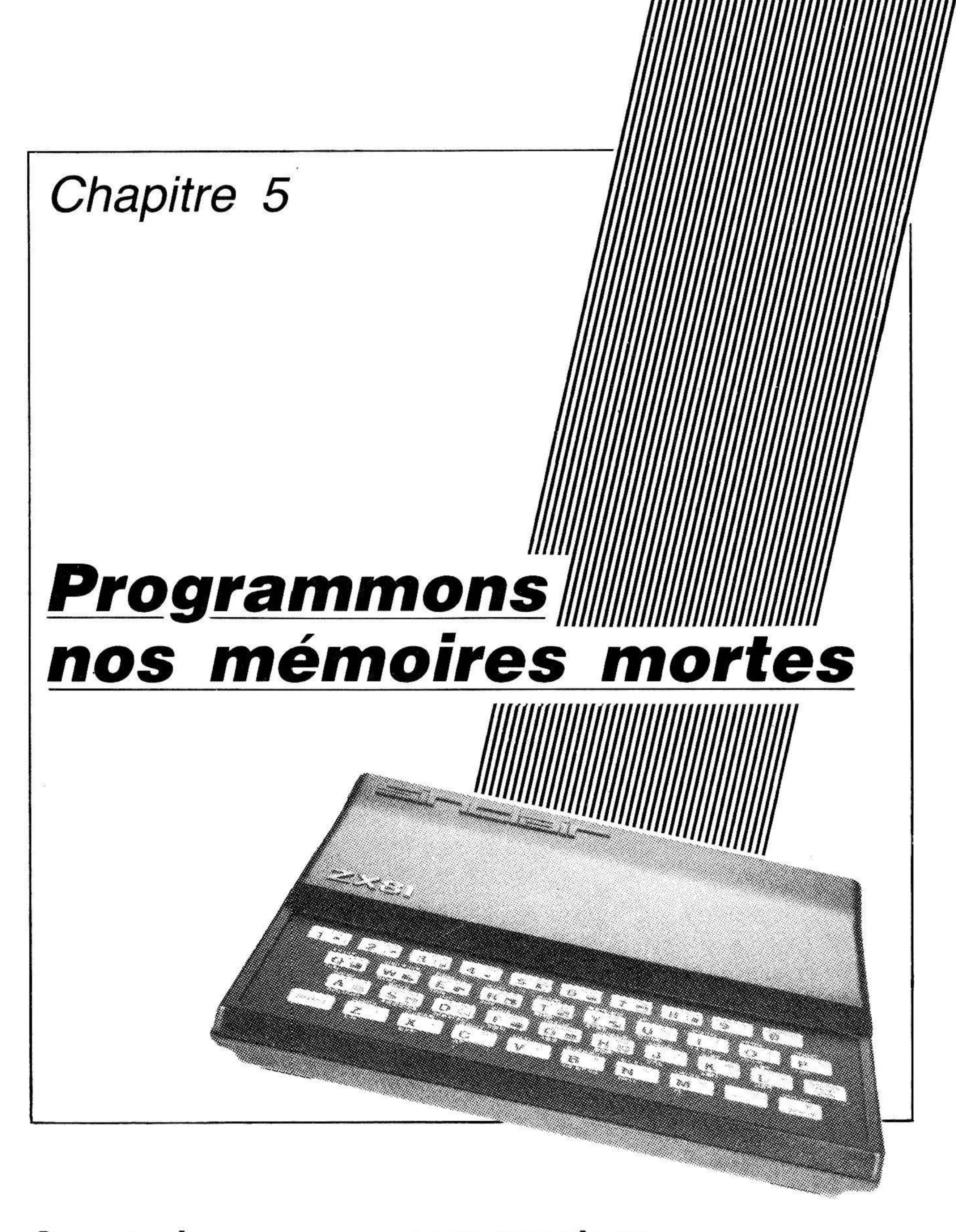
Avant de retirer la ROM d'origine, on repèrera bien son branchement car quatre des contacts de son support sont inutilisés. Notre adaptateur se monte bien sûr exactement de même.

---

(1) Voir photo de couverture.

Chapitre 5

Programmons nos mémoires mortes



Construisons un programmateur

Nous venons d'étudier deux possibilités permettant d'équiper le ZX 81 d'une EPROM 2716 programmée par l'utilisateur.

La *figure 5-1* montre maintenant quelles sont les fonctions que devra remplir le programmateur que nous allons réaliser. Il semble logique

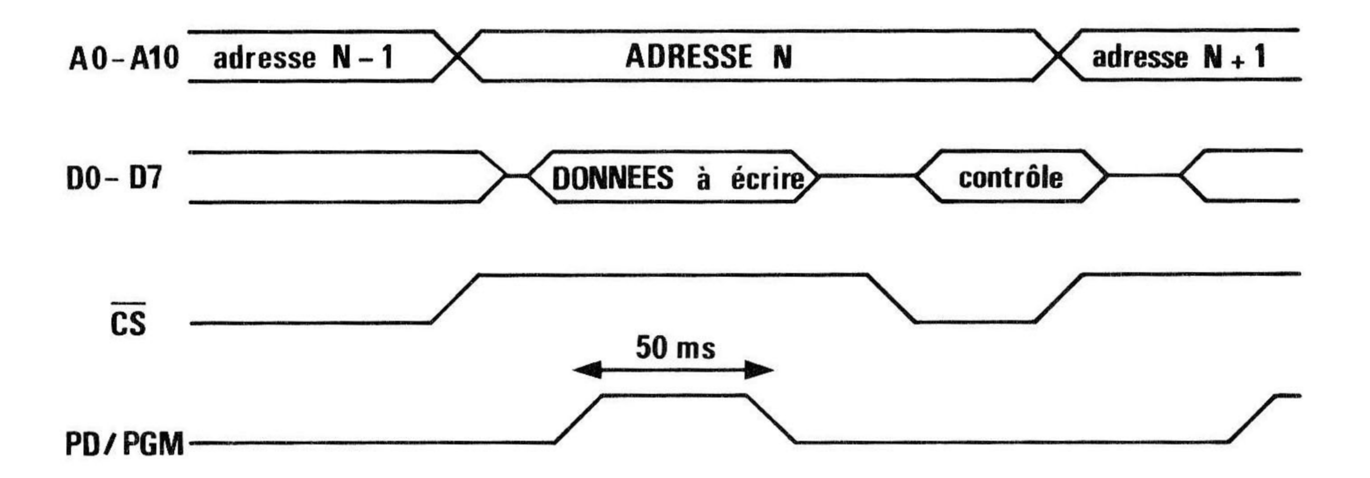


Fig. 5-1. — *N.B. Ce diagramme des temps est celui recommandé par les fabricants. Des variantes sont possibles, le contrôle étant, par exemple, facultatif.*

d'utiliser la puissance de traitement d'un ZX 81 pour simplifier au maximum les circuits nécessaires. C'est dans cette optique qu'a été étudié le schéma de la *figure 5-2* qui reporte sur le logiciel la plupart des fonctions tant soit peu critiques, notamment la génération de l'impulsion de programmation qui doit être parfaitement calibrée à 50 ms.

En fait, le circuit se compose essentiellement d'une batterie de « latches » de type 74LS75, chargés de retenir pendant tout le temps nécessaire les informations très fugaces fournies par les bus d'adresses et de données.

Comme seules les lignes AØ à A1Ø sont reconnues par la 2716 à programmer, il a été décidé d'utiliser la ligne A11 pour transmettre l'impulsion d'écriture, et les lignes A12 à A15 pour réaliser un adressage en mémoire dans une zone située au-delà d'une extension 16 K. Les possesseurs de boîtiers RAM de capacité supérieure veilleront simplement à éviter tout double emploi des localisations réservées au-dessus de 45 055.

Cette zone correspond en fait à l'une des multiples « réflexions » de la ROM, et il est donc indispensable de faire « place nette » en forçant $\overline{ROMCS}$ à 1 par l'intermédiaire d'une diode. Un 74LS00 et un 74LS30 suffisent à réaliser ces fonctions combinatoires très simples. $\overline{MREQ}$ et $\overline{WR}$ rentrent bien sûr aussi dans la composition du signal de validation des latches, puisqu'il n'est pas fait usage de ports d'entrée-sortie.

Il a été prévu une petite alimentation secteur pour le +25 V de programmation, mais trois piles miniature de 9 V en série pourront également être utilisées si le programmateur ne sert pas de façon par trop intensive.

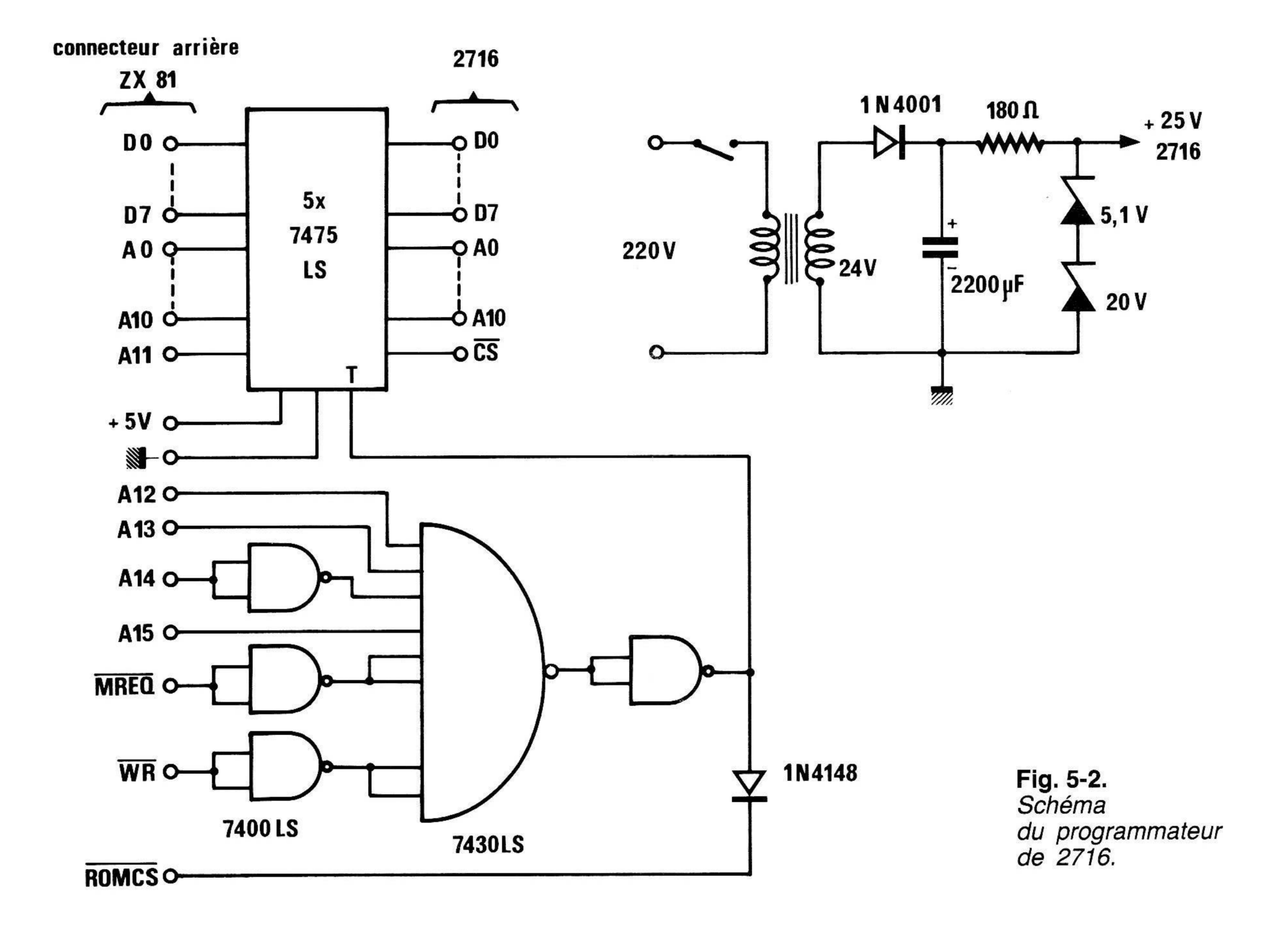


Fig. 5-2.
Schéma du programmateur de 2716.

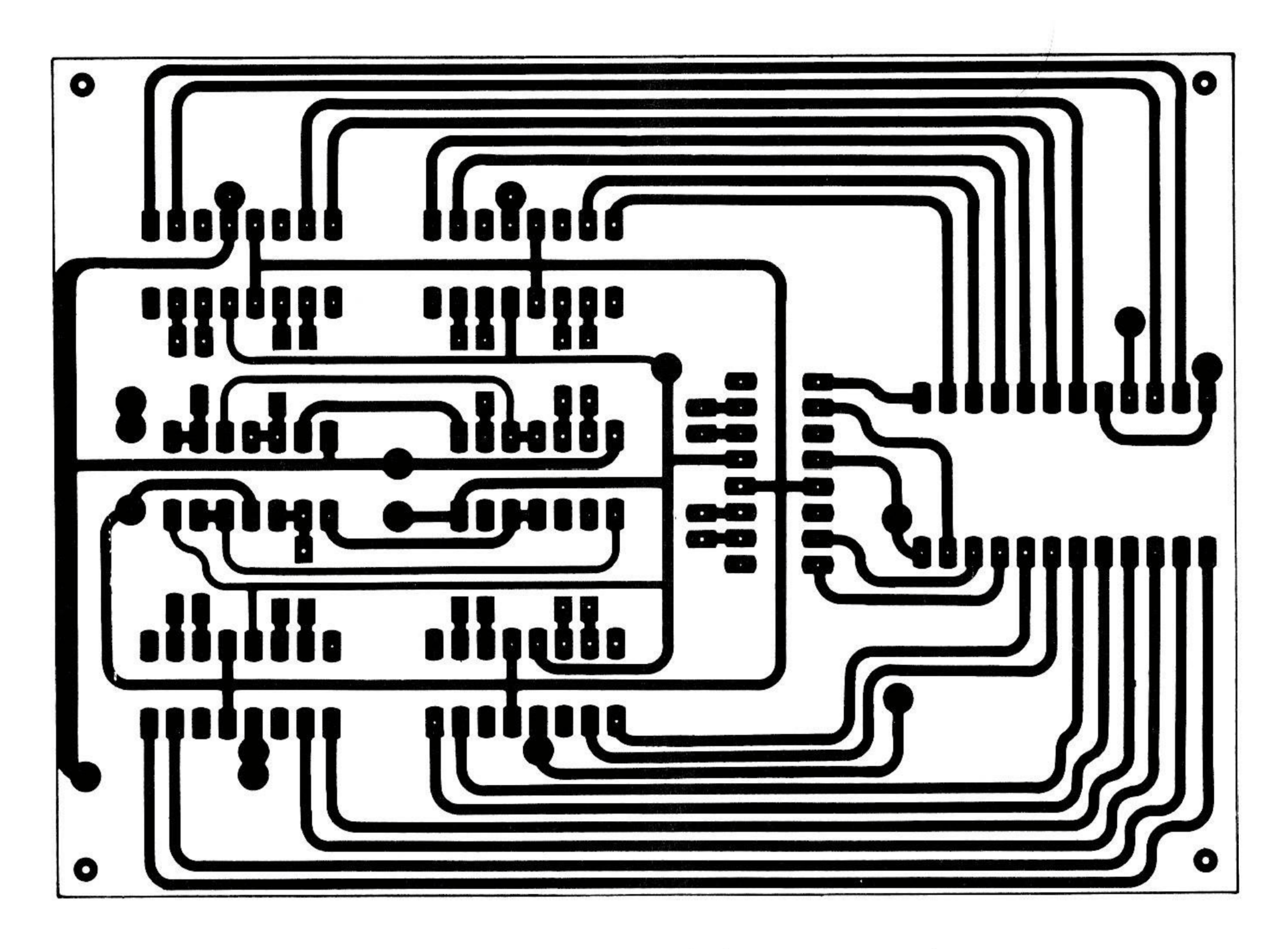


Fig. 5-3. - *Circuit imprimé du programmateur.*

Rappelons qu'en aucun cas cette « haute tension » ne doit être appliquée (ou rester appliquée) en l'absence du 5 V, sous peine de destruction immédiate de l'EPROM. On suivra donc scrupuleusement les ordres apparaissant sur l'écran TV, tout en s'abstenant de modifier les logiciels proposés.

La *figure 5-3* reproduit le tracé du circuit imprimé simple face destiné à accueillir tous les composants du montage selon le plan d'implantation de la *figure 5-4*. Il est important de ne pas oublier les cinq straps, et de bien veiller à l'orientation des circuits intégrés.

Sans aller jusqu'à préconiser le recours à un support à force d'insertion nulle pour la 2716, nous insisterons sur la nécessité de choisir un modèle d'excellente qualité, en raison de la fréquence des enfichages et désenfichages à venir.

L'alimentation secteur pourra être réalisée selon les plans des

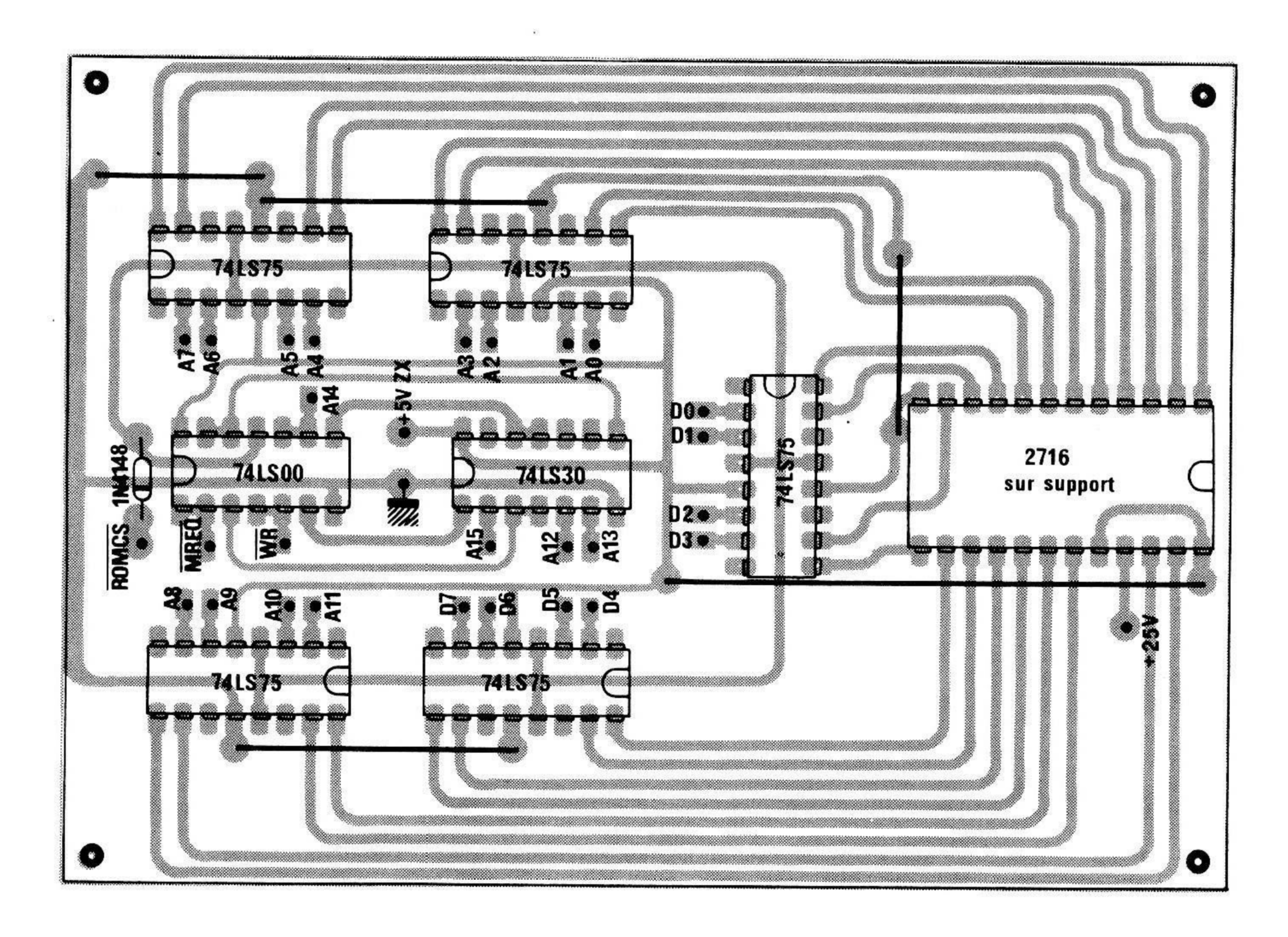


Fig. 5-4. - *Plan de câblage du programmateur.*

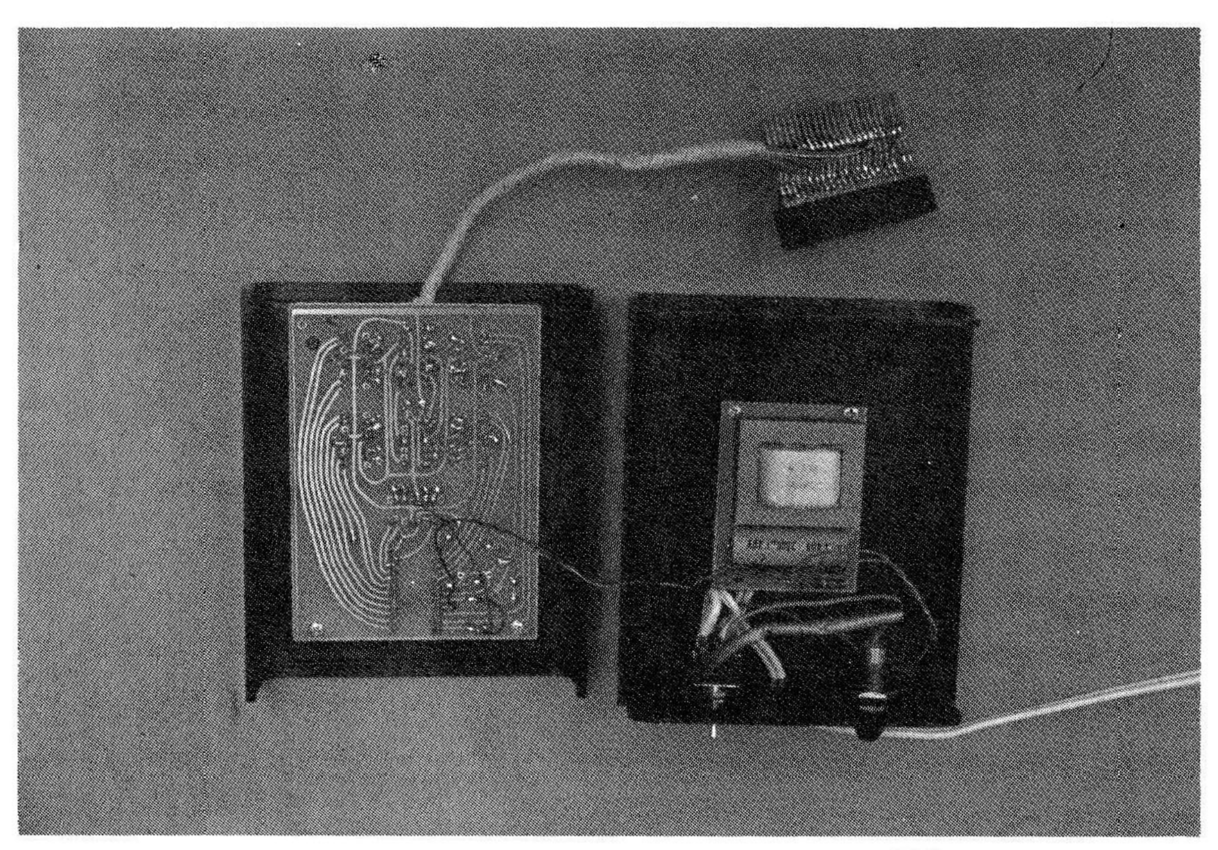
figures 5-5 et 5-6 à moins que la solution « piles » soit préférée (consommation moyenne de l'ordre de 10 mA en programmation).

Les dimensions des circuits permettent d'envisager leur montage dans un boîtier plastique noir 115 PP de MMP, dont l'esthétique s'accorde assez bien avec celle du ZX.

Il faut prévoir un connecteur de raccordement à 44 broches, avec partie mâle pour le branchement de l'extension mémoire.

La *figure 5-7* donne le détail de réalisation de cette pièce, tout en fournissant le repérage des connexions à opérer au moyen d'un toron de fils souples ou d'un morceau de câble en nappe (20 cm au maximum dans les deux cas).

Insistons sur la nécessité FONDAMENTALE d'un contrôle soigné, se référant au besoin au schéma « officiel » du ZX : toute erreur de branchement risquerait d'endommager tant le programmateur que l'ordinateur !



Vue interne du programmateur d'EPROM.

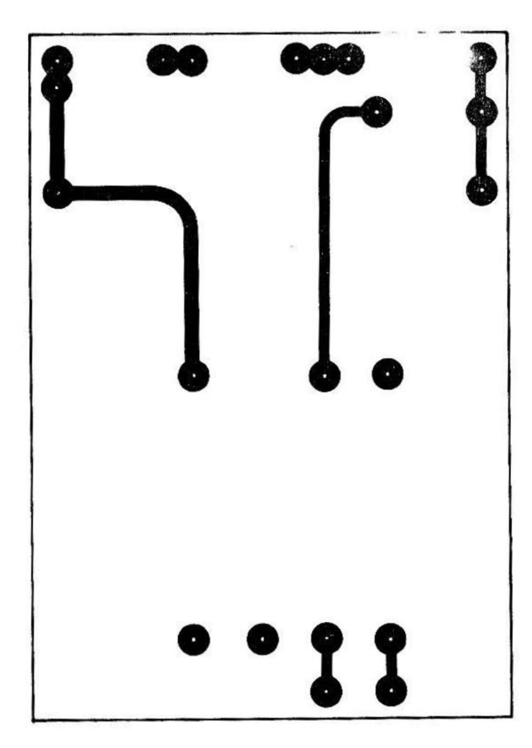


Fig. 5-5.
Circuit imprimé de l'alimentation.

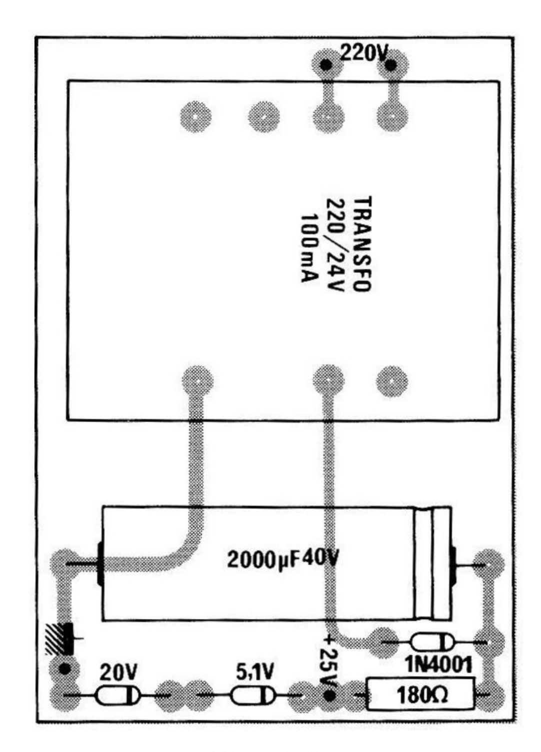
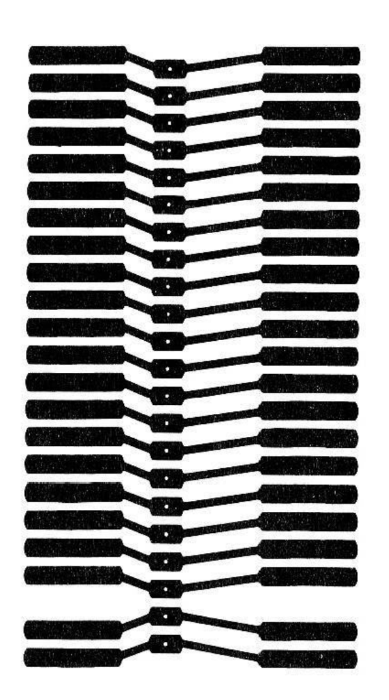
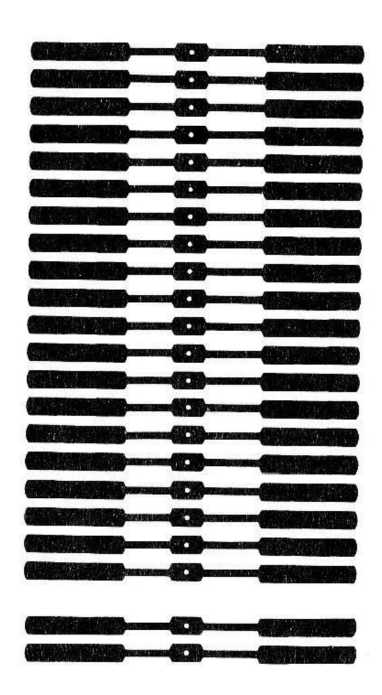
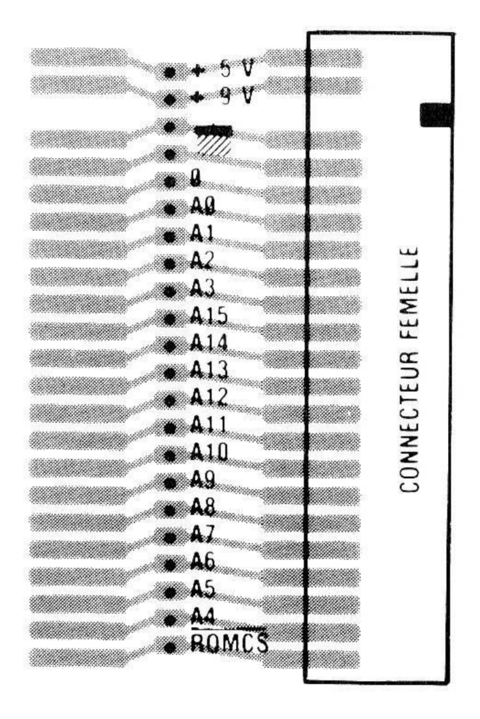


Fig. 5-6.
Implantation de l'alimentation.






+ 5 V
+ 9 V
0
A0
A1
A2
A3
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
ROMCS
CONNECTEUR FEMELLE

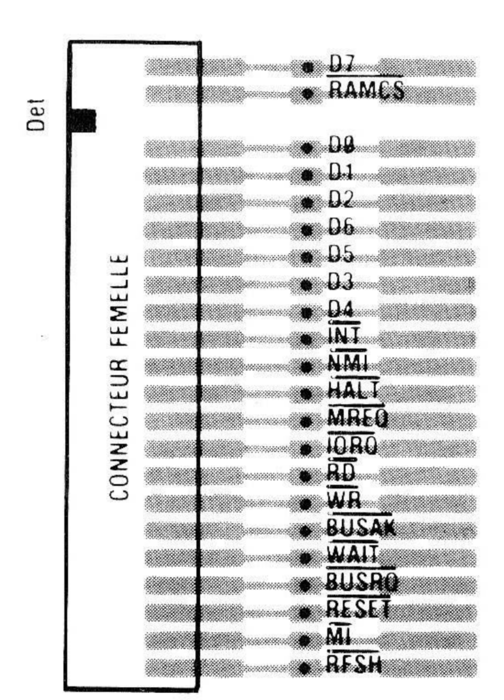

Det
CONNECTEUR FEMELLE
D7
RAMCS
D0
D1
D2
D6
D5
D3
D4
INT
NMI
HALT
MREQ
IORQ
RD
WR
BUSAK
WAIT
BUSRQ
RESET
MI
RFSH

Fig. 5-7.

Logiciels de programmation

Sans un logiciel de programmation, ce montage n'est pas plus utilisable qu'un ZX 81 sans ROM !

La *figure 5-8* fournit un programme simple assurant néanmoins toutes les fonctions nécessaires. Son principal avantage est de se contenter de peu de mémoire !

S'il devait être utilisé sans extension mémoire, il conviendrait de modifier sa ligne 100 en une PAUSE 2 au lieu de 1.

Ce logiciel charge en EPROM, à partir de l'adresse Ø, une suite de 2 048 octets au maximum, qui doivent être rangés, sous forme de caractères, dans une chaîne nommé A$. Bien sûr, s'il ne faut programmer qu'un nombre d'octets inférieur à 2 048, la machine s'arrêtera d'elle-même sur la ligne 70 lorsque la fin de la chaîne sera atteinte (compte rendu d'erreur).

```
  1 REM PROGRAMMATION 2716 16K
  5 PRINT "BRANCHER LE 25 VOLTS
  6 PRINT
  7 PRINT "PUIS PRESSER R"
 10 POKE 45056,255
 20 IF INKEY$="R" THEN GOTO 50
 30 GOTO 10
 50 CLS
 60 FOR F=1 TO 2048
 70 LET D=CODE A$(F)
 80 LET A=F-1
 90 POKE (45056+A),D
 95 POKE (47104+A),D
100 PAUSE 1
110 POKE (45056+A),D
120 PAUSE 10
130 NEXT F
140 PRINT "COUPER LE 25 VOLTS"
150 REM COPYRIGHT 1983
```

Fig. 5-8. — *Le logiciel du programmateur. Les octets à implanter sont supposés rangés dans A$. Avec 1K octet de RAM, faire 100 PAUSE 2. ATTENTION : le lancement doit se faire par GOTO 1 et non par RUN !*

Il existe différents moyens permettant de construire la chaîne A$. Le logiciel « de luxe » de la *figure 5-9* procure un maximum de confort et de sécurité, grâce à des possibilités de correction et de sauvegarde sur cassette avant mise en EPROM. De plus, pendant le transfert dans la 2716, tous les octets sont visualisés avec leurs adresses à des fins de vérification éventuelle.

```
   1 REM PROGRAMMATION 2716
   2 POKE 45056,255
   3 GOTO 200
   4 CLS
   5 PRINT "ALIMENTER LE PROGRAM
MATEUR"
   6 PRINT
   7 PRINT "PUIS PRESSER R"
  10 POKE 45056,255
  20 IF INKEY$="R" THEN GOTO 40
  30 GOTO 10
  40 POKE 45056,255
  50 CLS
  60 FOR F=1 TO LEN A$
  70 LET D=CODE A$(F)
  80 LET A=F-1
  90 POKE (45056+A),D
  95 POKE (47104+A),D
 100 PAUSE 1
 110 POKE (45056+A),D
 120 SCROLL
 122 PRINT A,D
 125 FOR G=0 TO 10
 127 NEXT G
 130 NEXT F
 132 SCROLL
 135 SCROLL
 140 PRINT "ARRETER LE PROGRAMMA
TEUR"
 145 SCROLL
 147 STOP
 150 PRINT "SAUVEGARDE CASSETTE
? O/N"
 160 IF INKEY$="N" THEN GOTO 4
 162 IF INKEY$="O" THEN GOTO 170
 165 GOTO 160
 170 SAVE "EPROM"
 180 GOTO 4
 200 LET A$=""
 210 LET L=0
 220 SCROLL
 230 PRINT L,
 240 INPUT B$
 250 IF B$="" THEN GOTO 300
 260 LET A$=A$+CHR$ VAL B$
 270 PRINT B$
 280 LET L=L+1
 290 GOTO 220
 300 SCROLL
 310 SCROLL
 320 SCROLL
 330 PRINT "****** ";LEN A$;" OC
TETS ENTREE ******"
 340 SCROLL
 350 SCROLL
 360 FOR F=0 TO 300
```

Fig. 5-9.
suite au verso

```
370 SCROLL
380 PRINT "CORRECT ? O/N"
390 IF INKEY$="O" THEN GOTO 520
400 IF INKEY$="N" THEN GOTO 420
410 GOTO 390
420 SCROLL
430 PRINT "ADRESSE A CORRIGER ?
"
440 INPUT A
450 SCROLL
460 PRINT A;
470 INPUT C
480 LET A$(A+1)=CHR$ C
490 PRINT C
500 SCROLL
510 GOTO 380
520 SCROLL
530 PRINT "TERMINE ? O/N"
540 SCROLL
550 IF INKEY$="O" THEN GOTO 150
560 IF INKEY$="N" THEN GOTO 220
570 GOTO 550
580 REM COPYRIGHT 1983
```

Fig. 5-9.

Dès son lancement par RUN, le programme demande l'octet à implanter à l'adresse Ø. Sitôt celui-ci introduit (en code décimal), une prise en compte est faite sur l'écran, et l'octet suivant est attendu.

On peut à tout instant arrêter la programmation en frappant seulement sur NEWLINE. Une possibilité de correction est alors offerte, après laquelle il est possible soit de continuer, soit de sauver la suite d'octets sur cassette, soit de transférer directement celle-ci dans l'EPROM, solution expéditive mais imprudente !

A cet instant ou après un rechargement de la cassette, l'ordre est donné d'appliquer le 25 V au programmateur. Après l'appui sur R (comme RUN), la programmation définitive s'effectue, avec contrôle permanent sur l'écran de l'avancement de l'opération : il faut compter douze minutes pour 2 048 octets, avec possibilité d'accélération au niveau de la ligne 125, aux dépens toutefois de la fiabilité du chargement...

Il est vital de débrancher le 25 V avant de mettre le ZX hors tension pour récupérer l'EPROM, et le message écran rappelle ce mode opératoire. Bien entendu, il est exclu de procéder à tout enfichage ou désenfichage tant de l'EPROM que du programmateur sur le ZX, en présence de l'une ou l'autre des tensions d'alimentation...

Conclusion

La réalisation de ce programmateur vient d'être décrite dans le cadre particulier de l'adaptation de mémoires EPROM sur le ZX 81. Il est cependant bien évident que cet appareil peut être utilisé pour la programmation de 2 716 destinées à des usages tout à fait différents. Dans de telles circonstances, on appréciera la souplesse d'emploi de l'ensemble qui a pu être obtenue grâce à une approche essentiellement logicielle du problème.

Le prix de revient de l'ensemble ZX 81 et programmateur est par ailleurs inférieur à celui de la plupart des programmateurs du commerce.

En complétant cet équipement par le petit effaceur UV dont nous introduirons la réalisation à la fin de ce chapitre, on disposera de moyens permettant d'exploiter entièrement les possibilités des mémoires non volatiles de grande diffusion que sont désormais les 2 716.

Des programmes BASIC dans des EPROMS

L'utilisation de cartouches embrochables permet à certaines machines (notamment aux ordinateurs de jeux) de disposer de toute une gamme de logiciels sans attente, dès la mise sous tension.

On a vu qu'il est facile d'adapter au ZX 81 de telles cartouches, équipées de mémoires mortes de type EPROM.

Toutefois, ces mémoires supplémentaires ne peuvent normalement accueillir que des programmes écrits en langage machine et appelés au moyen de la fonction USR.

Nous allons voir ici comment procéder pour loger dans ces EPROMS, des programmes BASIC auxquels l'accès sera tout aussi instantané.

RUN contre RAND USR

Le ZX 81 est une machine normalement conçue pour exécuter un unique programme BASIC présent en mémoire, et éventuellement des routines écrites en langage machine.

La structure de l'ordinateur veut que le programme BASIC soit logé, en mémoire, à partir de l'adresse 16509 et que le fichier d'affichage le suive immédiatement. Aucun programme BASIC ne pourra être lancé (par RUN ou GOTO) s'il n'occupe pas cette place réservée.

A l'inverse, un programme en langage machine peut être implanté n'importe où en mémoire pourvu qu'il ne perturbe pas, par sa présence, le fonctionnement normal de l'ordinateur.

Lorsque l'on ajoute des EPROMS dans l'espace mémoire d'origine, il n'est évidemment pas question de les adresser à partir de 16 509 puisque cette zone fait l'objet de constants mouvements d'octets, gérés par le moniteur Sinclair. Le plus souvent d'ailleurs, on choisit d'implanter les EPROMS additionnelles entre 8 192 et 16 383, c'est-à-dire dans l'espace laissé libre par les concepteurs de la machine entre la ROM et la RAM.

Il ne pourra donc jamais être question de lancer directement un programme BASIC logé dans une EPROM, alors que la chose est parfaitement habituelle pour du code machine (en faisant, par exemple, RAND USR 8192).

Si on tient néanmoins à recopier un programme BASIC existant, dans une EPROM, alors il faudra impérativement le *transférer* à l'adresse 16509 avant de pouvoir le lancer par un RUN ou un GOTO. Même chose pour les variables qui pourraient éventuellement l'accompagner et qui devront être transférées dans la zone de la mémoire qui leur est réservée.

De tels transferts n'ont rien de nouveau ; on en fait déjà usage dans bien des cas, dont voici quelques exemples :

— stockage temporaire au-dessus de RAMTOP d'un programme BASIC pendant qu'on en charge un second, à partir de la cassette, pour fusionner les deux logiciels en un seul, plus long (logiciel PROGMERGE de ACS Software) ;
— stockage au-dessus de RAMTOP d'un programme compilateur pendant qu'on charge le logiciel à compiler (MCODER de PSS) ;
— mise à l'abri de programmes BASIC dans une extension mémoire permanente (technologie CMOS-pile au lithium). Cette dernière application n'est d'ailleurs pas sans rappeler notre propos... (mémoires CAMEL et SAM).

Confiés au BASIC, ces transferts seraient affreusement longs, et même impossibles dans le sens du rechargement : le programme à charger viendrait « écraser » le programme chargeur !

C'est donc au langage machine que l'on confie ce travail et, plus précisément, à la très puissante instruction LDIR de l'assembleur du Z 80.

Les merveilles de LDIR

L'instruction LDIR faite partie du jeu d'instructions de transfert par blocs de l'assembleur du Z 80 et n'existe guère que sur ce microprocesseur très performant qui a été fort à propos choisi par Sinclair pour équiper le ZX 81.

Cette instruction remplace à elle seule une routine entière, puisqu'il suffit de charger la première adresse du bloc d'octets à transférer dans le registre HL, l'adresse du premier octet de la zone devant recevoir le bloc dans le registre DE, et le nombre d'octets du bloc dans le registre BC, puis d'exécuter LDIR pour que le transfert s'effectue en un temps record. Moins d'une seconde suffit à déplacer tout le contenu d'une mémoire 16 K!

La *figure 5-10* donne la liste des vingt-quatre octets constituant deux routines accomplissant deux tâches complémentaires :

- recopie de toute la mémoire 1K (moins 24 octets) d'un ZX 81 de base, dans une zone mémoire logée de 8 216 à 9 215;
- recopie des 1 000 octets compris entre 8 216 et 9 215, à partir de 16 384.

Fig. 5-10.

16514	1
16515	232
16516	3
16517	17
16518	24
16519	32
16520	33
16521	0
16522	64
16523	237
16524	176
16525	201
16526	1
16527	232
16528	3
16529	17
16530	0
16531	64
16532	33
16533	24
16534	32
16535	237
16536	176
16537	201

Expliquons-nous :

La première routine va recopier tout le contenu de la RAM (programme et variables, mais aussi variables système et pile machine) à partir de l'adresse 8216. Il faut, bien sûr, que cette zone soit équipée de mémoire RAM, ce qui peut être obtenu par les moyens suivants :

- utilisation d'une extension 64 K;
- utilisation de certaines cartes génératrices de caractères incorporant

de la RAM à cet endroit (par exemple le module HIREZ de DIDECAR Marketing qui, hélas, n'est plus guère disponible).

- en présence d'une extension 16 K, remise en service du boîtier 1 K d'origine, qui se trouve normalement neutralisé (voir notre montage du chapitre 4).

En cas de doute, il suffit, *sur le système complet,* de lancer le court programme de la *figure 5-11* pour obtenir, dans les minutes qui suivent, un jugement sans appel.

```
  10 FOR F=8192 TO 9215
  20 POKE F,0
  30 IF PEEK F<>0 THEN GOTO 100
  40 POKE F,255
  50 IF PEEK F<>255 THEN GOTO 10
0
  60 NEXT F
  70 PRINT AT 10,0;"CE SYSTEME C
ONVIENT"
  80 STOP
  90 REM COPYRIGHT 1983
 100 PRINT AT 10,0;"CE SYSTEME N
E CONVIENT PAS"
```

Fig. 5-11.

Une solution de rechange consisterait à modifier les routines de la *figure 5-10* de façon à ce que le programme soit stocké au-dessus de RAMTOP : avec une extension 16, 32, ou 64 K, la chose est facile, puisqu'il suffit de lancer un NEW précédé d'un POKE 16389,68 pour que la machine « redevienne une 1 K », mais avec 15 K-octets disponibles à partir de 17 340.

Cette manœuvre sera de toute façon indispensable, même en présence de l'un des cas précédents, puisque nous n'envisageons ici, pour des raisons évidentes de coût, que le chargement en EPROM de programmes 1 K RAM. Rappelons que de l'excellent travail peut être fait avec 1 K de BASIC, comme notre confrère G. Isabel (1) l'a largement prouvé !

Ceux de nos lecteurs qui souhaiteraient, pour ce motif ou pour toute autre raison, modifier ces routines, trouveront en *figure 5-12* un désassemblage hexadécimal qui leur facilitera grandement le travail !

Cette liste montre que ces vingt-quatre octets sont implantés à partir de l'adresse 8182 (2 000 h), c'est-à-dire tout au début de la zone de RAM qui servira à accueillir le BASIC (d'où la perte de 24 octets déjà signalée).

---

(1) *« 50 programmes pour ZX 81 »,* Poche-informatique, n° 1, ETSF.

```
*D
=2000
2000 LD   BC,03E8
2003 LD   DE,2018
2006 LD   HL,4000
2009 LDIR
200B RET
200C LD   BC,03E8
200F LD   DE,4000
2012 LD   HL,2018
2015 LDIR
2017 RET
Q
*G
```

Fig. 5-12.

```
1 REM ABCDEFGHIJKLMNOPQRSTUVWX
2 FOR F=16514 TO 16514+23
3 INPUT C
4 POKE F,C
5 NEXT F
6 REM COPYRIGHT 1983
```

Fig. 5-13.

Cette place un peu insolite a été choisie afin que le programme BASIC et ces deux routines indispensables forment un tout indissociable. Ainsi, même après le transfert en EPROM de tout le bloc de 1 024 octets, la routine de transfert sera immédiatement disponible sans autre manœuvre que l'embrochage de la « cartouche ».

Avant de passer à la programmation de véritables EPROMS, il semble commode de procéder à une sorte de « répétition » sur de la RAM :

On commencera donc par charger le court programme de la *figure 5-13* sur une machine 1 K RAM. Une fois ce logiciel lancé, on frappera, en les séparant par NEWLINE, les vingt-quatre octets dont la liste est donnée à la *figure 5-10.*

La ligne 1 du programme revêt à présent l'aspect représenté sur la *figure 5-14.* On frappera alors les lignes 2 à 9 pour achever la construction du programme définitif.

Après un listage de contrôle, on déclenchera une sauvegarde sur cassette grâce à un GOTO 7. NE PAS faire RUN auparavant, car la ligne 6 effacerait la mémoire !

Pour transférer un programme BASIC à l'adresse 8216, on chargera la cassette précédemment enregistrée, *sur le système complet,* préalablement testé au moyen du programme de la *figure 5-11.*

```
   1 REM ▪  CONT ▀) /45 RND GOSUB
█TAN ▪ CONT ▀) RND5/4 GOSUB █TAN

   2 FOR F=1 TO 24
   3 POKE 8191+F,PEEK (16513+F)
   4 NEXT F
   5 POKE 16389,68
   6 NEW
   7 SAVE "LDI█"
   8 RUN
   9 REM COPYRIGHT 1983
```

Fig. 5-14.

```
   1 REM PROGRAMMATION 2716
   2 REM EN BASIC
   3 POKE 45056,1
   5 PRINT "ALIMENTER LE PROGRAM
MATEUR"
   6 PRINT
   7 PRINT "PUIS PRESSER R"
  10 POKE 45056,1
  20 IF INKEY$="R" THEN GOTO 40
  30 GOTO 10
  40 POKE 45056,1
  50 CLS
  60 FOR F=1 TO 1024
  70 LET D=PEEK (F+8191)
  80 LET A=F-1
  90 POKE (45056+A),D
  95 POKE (47104+A),D
 100 PAUSE 2
 110 POKE (45056+A),D
 122 PRINT AT 10,0;A;"      ",D;"
 "
 125 FOR G=0 TO 10
 127 NEXT G
 130 NEXT F
 140 PRINT AT 0,0;"ARRETER LE PR
OGRAMMATEUR"
 180 STOP
 200 SAVE "EPRO█"
 205 RUN
 210 REM COPYRIGHT 1983
```

Fig. 5-15.

Le chargement de la cassette (par LOAD « ») ne renvoie pas le compte rendu ØØ habituel, mais le curseur K au bout d'un léger temps d'attente. Non, le chargement n'a pas échoué, simplement le NEW de la ligne 6 a bien rempli son rôle. On peut alors charger le programme BASIC 1 K RAM à transférer, de façon tout à fait normale.

Cela fait, il suffira de lancer un RAND USR 8192 pour que le transfert s'effectue en une fraction de seconde.

Exécutons un NEW afin de détruire l'original du programme BASIC : il ne reste plus qu'à « rappeler » ce dernier par un RAND USR 8204. Dès l'apparition du compte rendu Ø/Ø, on peut lancer un LIST, le programme qui vient d'être effacé doit à nouveau être présent en machine !

Il est intéressant de comparer la vitesse d'un tel rechargement à celle d'un LOAD depuis une cassette...

Il ne reste plus qu'à étendre la procédure à la programmation d'une EPROM pour atteindre le but annoncé : le chargement instantané de programmes BASIC par cartouche.

Programmation d'une EPROM en Basic

Pour des raisons qui ont déjà été expliquées, nous avons retenu les EPROMS de type 2716. Il s'agit de mémoires de 2 K-octets, ce qui signifie que, moyennant une très légère adaptation des logiciels fournis ici, il sera possible de loger un second programme BASIC 1 K RAM à la suite du premier. Les deux programmes seront accessibles tout aussi instantanément par des RAND USR distincts.

La programmation nécessite... un programmateur, par exemple celui dont nous avons décrit la réalisation plus avant. Cet accessoire étant en place, il suffit de procéder aux opérations précédentes, mais au lieu de rappeler le programme BASIC en fin de transfert, on chargera le logiciel de la *figure 5-15*. Une fois lancé, il recopiera intégralement dans l'EPROM le programme d'origine, accompagné de ses routines de transfert.

Reste maintenant à adapter cette EPROM sur le ZX 81 chaque fois que le programme qu'elle contient devra être utilisé. Il est possible d'employer tous les adaptateurs (et ils sont nombreux !) capables d'adresser l'EPROM à partir de l'adresse 8192, et tout particulièrement le premier décrit dans notre chapitre 4. En aucun cas cette EPROM ne devra être montée en remplacement de la ROM Sinclair puisque l'exécution d'un programme BASIC nécessite l'interpréteur ! L'adaptateur (muni de l'EPROM) étant embroché, il suffit de mettre la machine sous tension et de lancer un RAND USR 8204 pour que le programme se charge instantanément.

On comprend alors la raison du choix d'une zone de RAM commençant à 8192 pour stocker temporairement le programme BASIC : les adresses sont alors les mêmes que celles de l'EPROM ! Comme nous l'avons signalé, d'autres choix sont possibles, notamment au-dessus de

RAMTOP, mais au prix d'une complication des procédures qui ont été décrites. D'autres modifications permettraient également de figer en EPROM des programmes BASIC de plus de 1 K-octet, mais le coût des mémoires 2732, 2764, et autres restreint sérieusement le domaine d'application du procédé pour les longs logiciels. De toute façon, dans le cadre de cet ouvrage, les programmes susceptibles de subir ce traitement seront essentiellement des logiciels de gestion des entrées-sorties, généralement assez courts.

Un programme... qui programme en langage machine

Il est bien certain qu'un tel programme ne se maintiendra dans des limites acceptables de simplicité que si son domaine d'intervention est parfaitement délimité. Le programme « miraculeux » n'existe évidemment pas, et il faudrait disposer d'une série complète de logiciels de ce type pour être en mesure de résoudre tous les cas de figure. Cependant, pourquoi pas ?

Nous nous limiterons ici à la description d'un logiciel capable d'élaborer un programme permettant à l'ensemble ZX 81-carte 8ES d'imposer à ses huit sorties un cycle prédéfini. Bien sûr, toutes les combinaisons marche-arrêt des sorties seront autorisées, ce qui offre 256 possibilités distinctes. Les changements d'état des sorties seront séparés par des temporisations programmées dans une fourchette s'étendant d'une seconde à 255 jours (à condition qu'aucune panne d'alimentation ne vienne interrompre le cycle !).

La capacité de la mémoire 2716 permet d'enchaîner une soixantaine de séquences, ce qui conduit à une durée maximale du cycle complet de près de 42 ans... Si cela ne suffit pas, le cycle peut reprendre au début lorsqu'il est achevé !

Bien sûr, de telles durées ne présentent aucun intérêt pour l'utilisateur amateur. Cependant, la porte est ouverte à des programmations à l'échelle de la saison, voire de l'année, de mises en service d'équipements extrêmement divers. On peut songer à l'automatisation de chauffages de résidences secondaires, d'aquariums, de systèmes d'alarme ou de simulation de présence, mais aussi de projecteurs de diapositives, de magnétophones, ou de jeux de lumière pour les animations les plus diverses !

C'est en définitive l'imagination du lecteur qui représente la seule limite en matière d'applications possibles...

```
   1 PRINT AT 10,3;"INITIALISATI
ON EN COURS"
   5 LET A$=""
  10 LET B$="0460010380010300010
06183014255022255021032253013032
24800503224302903223803703223304
5032228"
  20 LET C$=""
  30 FOR F=1 TO 30
  40 LET C$=C$+CHR$ VAL B$(3*F-2
 TO 3*F)
  50 NEXT F
  60 LET S=0
  70 DIM S(8)
  75 CLS
  80 PRINT AT 5,0;"ETAT DES SORT
IES:"
  90 PRINT AT 8,5;"1   2   3   4   5
6   7   8"
 100 PRINT AT 10,5;
 110 FOR F=1 TO 8
 120 PRINT S(F);"   ";
 130 NEXT F
 140 PRINT AT 15,0;"NUMERO DE SO
RTIE A INVERSER ?"
 145 PRINT AT 17,0;"(SINON,PRESS
ER NEWLINE SEULE)"
 150 INPUT N$
 160 IF N$="" THEN GOTO 300
 170 CLS
 175 PRINT AT 5,0;"ATTENDEZ..."
 180 LET N=VAL N$(1)
 190 IF S(N)=1 THEN GOTO 220
 200 LET S(N)=1
 210 GOTO 230
 220 LET S(N)=0
 230 LET S=0
 240 FOR F=1 TO 8
 250 LET S=S+(S(F)*(2**(F-1)))
 260 NEXT F
 270 PRINT AT 14,8;"ENCORE UN"
 280 GOTO 80
 300 LET A$=A$+CHR$ 62+CHR$ S+CH
R$ 211+CHR$ 127
 310 CLS
 320 PRINT AT 5,0;"TEMPORISATION
 EN:"
 330 PRINT AT 8,0;"SECONDES, PRE
SSER S"
 340 PRINT AT 10,0;"MINUTES, PRE
SSER M"
 350 PRINT AT 12,0;"HEURES, PRES
SER H"
 360 PRINT AT 14,0;"JOURS, PRESS
ER J"
 370 PRINT AT 20,5;"PUIS NEWLINE
```

Fig. 5-16.
suite au verso

```
 380 INPUT M$
 390 CLS
 400 PRINT AT 10,5;"DUREE TEMPO
EN ";M$;" ?"
 405 PRINT AT 12,10;"MAXI 255"
 410 INPUT D
 415 IF D>255 THEN GOTO 410
 420 LET D$=C$
 430 IF M$="S" THEN GOTO 500
 440 IF M$="M" THEN GOTO 600
 450 IF M$="H" THEN GOTO 700
 460 IF M$="J" THEN GOTO 800
 470 GOTO 300
 500 LET D$(8)=CHR$ (3+INT D)
 510 GOTO 1000
 600 LET D$(6)=CHR$ INT D
 610 GOTO 1000
 700 LET D$(4)=CHR$ INT D
 710 LET D$(6)=CHR$ 61
 720 GOTO 1000
 800 LET D$(2)=CHR$ INT D
 810 LET D$(4)=CHR$ 25
 820 LET D$(6)=CHR$ 61
 830 GOTO 1000
1000 LET A$=A$+D$
1010 CLS
1015 IF LEN A$>=2000 THEN PRINT
AT 5,0;"ATTENTION,DERNIERE PHASE
"
1020 PRINT AT 10,0;"CYCLE TERMIN
E ?  O/N"
1030 IF INKEY$="N" THEN GOTO 75
1040 IF INKEY$="O" THEN GOTO 106
0
1050 GOTO 1030
1060 CLS
1070 PRINT AT 10,0;"RETOUR AU DE
BUT (R) OU "
1080 PRINT AT 12,0;"ARRET DEFINI
TIF (A) ?"
1090 IF INKEY$="R" THEN GOTO 112
0
1100 IF INKEY$="A" THEN GOTO 120
0
1110 GOTO 1090
1120 LET A$=A$+CHR$ 195+CHR$ 0+C
HR$ 32
1130 GOTO 1300
1200 LET A$=A$+CHR$ 201
1300 CLS
1305 PRINT AT 10,6;"ATTENDEZ..."
1310 FOR F=1 TO LEN A$
1320 POKE 24999+F,CODE A$(F)
1330 NEXT F
1331 CLS
1333 PRINT AT 10,0;"POUR LISTER
PRESSER K"
```

```
1335 PRINT AT 12,0;"POUR ESSAYER
 SUR SES PRESSER R"
1337 IF INKEY$="K" THEN GOTO 135
0
1339 IF INKEY$="R" THEN GOTO 134
2
1340 GOTO 1337
1342 FAST
1344 RAND USR 25000
1345 SLOW
1346 GOTO 1331
1350 DIM E(8)
1355 CLS
1360 LET L=1
1362 SCROLL
1365 LET IN=CODE A$(L)
1370 LET W=128
1380 FOR Q=1 TO 8
1390 LET E(9-Q)=INT (IN/W)
1400 IF IN>=W THEN LET IN=IN-W
1410 LET W=W/2
1420 NEXT Q
1430 PRINT L,
1440 FOR F=8 TO 1 STEP -1
1450 PRINT E(F);
1460 NEXT F
1470 LET L=L+1
1480 SCROLL
1485 INPUT Z$
1490 GOTO 1365
1500 REM COPYRIGHT 1983
```

Fig. 5-16. — *Le programme Basic.*

La *figure 5-16* reproduit la liste complète du programme BASIC, destiné à fonctionner sur un ZX 81 équipé d'au moins 16 K de RAM.

Le dialogue qui s'établit comporte deux phases principales qui s'enchaîneront sans relâche :

— acquisition des modifications à apporter aux états des sorties après rappel de l'état présent ;
— acquisition de la durée de la temporisation à respecter avant la prochaine action sur les sorties, cette durée pouvant être indiquée en secondes, minutes, heures ou jours.

La combinaison directe de ces trois unités n'est pas possible, mais rien n'empêche d'enchaîner, sans modification de l'état des sorties, plusieurs temporisations utilisant des unités différentes. Egalement, chaque durée peut atteindre un maximum de 255 unités, ce qui offre d'intéressantes possibilités de recouvrement : 3 heures 6 minutes pourra ainsi être programmé sous la forme 186 minutes.

Toutes les durées sont dérivées d'une « base de temps » interne d'une seconde, obtenue à partir de la fréquence d'horloge. La précision obtenue sera donc liée à celle du filtre céramique, et à celle de l'étalonnage prévu à la *figure 5-17.* Ces deux lignes de programme permettent de « régler » le logiciel en fonction de la fréquence exacte, en MHz, de l'horloge du système.

```
54 LET QUARTZ=2.5
55 LET C$(10)=CHR$ INT (255*QU
ARTZ/3.25)
```

Fig. 5-17. — *Lignes à ajouter si le microprocesseur est équipé d'un quartz de fréquence autre que 3,25 MHz, cas de la carte du chapitre 9.*

A défaut de ces deux lignes, le logiciel utilise d'office une valeur de 3,25 MHz.

En fin de cycle, une dernière option est proposée :

— arrêt définitif, auquel cas le dernier état pris par les sorties subsistera jusqu'à la mise hors tension générale ;
— retour au début, ce qui fera boucler le cycle sur lui-même jusqu'à la mise hors tension générale.

En cours de dialogue, et spécialement à la fin, on remarquera des périodes d'attente plus ou moins longues : c'est pendant ces périodes que le ZX 81 écrit le langage machine prenant en compte les dernières informations acquises.

Nous ne fournirons aucune explication quant au fonctionnement de ce programme : en effet, celles-ci feraient perdre tout intérêt à l'écriture automatique de langage machine puisqu'il suffirait d'appliquer à la main les règles utilisées par ce logiciel !

Nos lecteurs habitués à la programmation en assembleur noteront sans doute d'affreuses redondances dans le code écrit par la machine. En fait, celles-ci ont été introduites sciemment, afin de faire l'économie de toute la RAM du système, qui pourra être carrément supprimée. L'élimination de ces lourdeurs passerait en effet par un recours à des instructions CALL, qui exigent une pile machine en RAM.

Transfert du logiciel sur EPROM (remplaçant la ROM Sinclair)

Au terme de l'écriture du code machine par le ZX 81, les octets du programme sont présents deux fois en machine : une première fois dans une chaîne A$, et une seconde fois à partir de l'adresse décimale 25000.

Cet apparent gaspillage d'espace mémoire ne coûte rien, mais améliore l'efficacité du programme :

En effet, l'utilisateur dispose de deux possibilités à la fin des opérations, voire même de trois.

La principale est l'édition sur écran (ou sur imprimante, au moyen d'une très simple adaptation) d'une liste binaire (poids faibles à droite) du logiciel Z 80. Une telle liste permet un transfert manuel immédiat des octets sur un programmateur simplifié. L'utilisateur devra presser NEWLINE chaque fois qu'il voudra faire avancer la liste d'un cran, ce qui lui laisse tout le temps souhaitable pour manœuvrer l'appareil. Une possibilité annexe consiste à lancer le programme, pour essai, directement sur le ZX 81 *équipé d'une carte d'entrée-sortie* de type 8ES. Deux remarques s'imposent cependant :

— cette possibilité ne doit pas être utilisée sur les programmes munis d'un retour au début automatique, car la machine ne pourrait plus revenir au BASIC et le programme serait perdu ;
— les durées des temporisations seront faussées, lors de cet essai, dans le rapport de 3,25 MHz à la fréquence déclarée à la ligne 54, si les deux lignes de la *figure 5-17* ont bien été prévues.

L'éventuelle troisième possibilité concerne ceux de nos lecteurs ayant réalisé notre programmateur d'EPROM adaptable au ZX 81 : le logiciel de cet accessoire devrait alors aller chercher les octets à programmer à partir de l'adresse 25000.

Conclusion

Quel que soit le procédé de programmation utilisé, cette EPROM peut être adaptée au ZX 81 soit à partir de notre adaptateur interne (avec ou sans RAM, mais à la place de la ROM Sinclair), soit au moyen de notre adaptateur externe (ROM Sinclair et RAM 1 K en place), soit sur la carte micro-processeur compatible qui sera décrite au chapitre 9.

Dans le premier et le dernier cas, le cycle prévu doit s'exécuter dès la mise sous tension du système. En cas de coupure secteur, le cycle reprendra au début, sans intervention. Dans le second cas, un lancement manuel par RAND USR 8192 est nécessaire.

Bien des fonctions domestiques peuvent ainsi être commandées, du chauffage central jusqu'à l'aquarium, avec une entière liberté de modification du cycle par simple effacement de l'EPROM.

Construction d'un effaceur d'EPROM

L'un des atouts essentiels des mémoires EPROM est la facilité avec laquelle elles peuvent être effacées, puis reprogrammées. L'effacement est obtenu en exposant la petite fenêtre du boîtier à un rayonnement ultra-violet.

La durée d'exposition dépend très largement de la nature de la source employée et de la distance à laquelle est placée la mémoire. Notons qu'une proximité exagérée risque d'entraîner un échauffement excessif de l'EPROM, pouvant déboucher sur sa destruction. Une distance de trois centimètres environ convient en général fort bien. Avec une source adéquate, il faut compter entre quinze et vingt minutes pour que tous les bits de la mémoire reviennent à un.

Avant tout réemploi, on pourra s'assurer par l'exécution d'un court programme BASIC, que tous les octets, au nombre de 2 048, sont bien revenus à 255.

Il existe dans le commerce des kits d'effaceurs, mais il est possible de commander chez la plupart des électriciens des *tubes germicides* du type TG 15 (15 watts).

Ces tubes sont analogues aux classiques tubes fluorescents, à ceci près que le revêtement interne est supprimé, afin de laisser passer le seul rayonnement de la vapeur de mercure. Il faut brancher ces tubes au secteur à travers une sorte de self appelé *ballast,* et leur adjoindre un *starter.* On veillera à bien respecter le schéma de câblage fourni car toute inversion de fils empêcherait l'allumage du ou des tubes. Il faut éviter de regarder ces tubes, lorsqu'ils sont allumés, sans lunettes de soleil, leur rayonnement étant nocif. La principale fonction du boîtage de l'effaceur sera donc de masquer le tube tout en permettant son aération. Les mémoires à effacer seront piquées dans une bande de mousse noire, récupérée dans un emballage pour circuits intégrés. A défaut, on pourra se rabattre sur du polystyrène expansé, mais IMPERATIVEMENT enveloppé de feuille d'aluminium ménager. A défaut, en effet, d'un support conducteur, si possible relié à la terre, les mémoires pourraient se trouver endommagées par l'électricité statique.

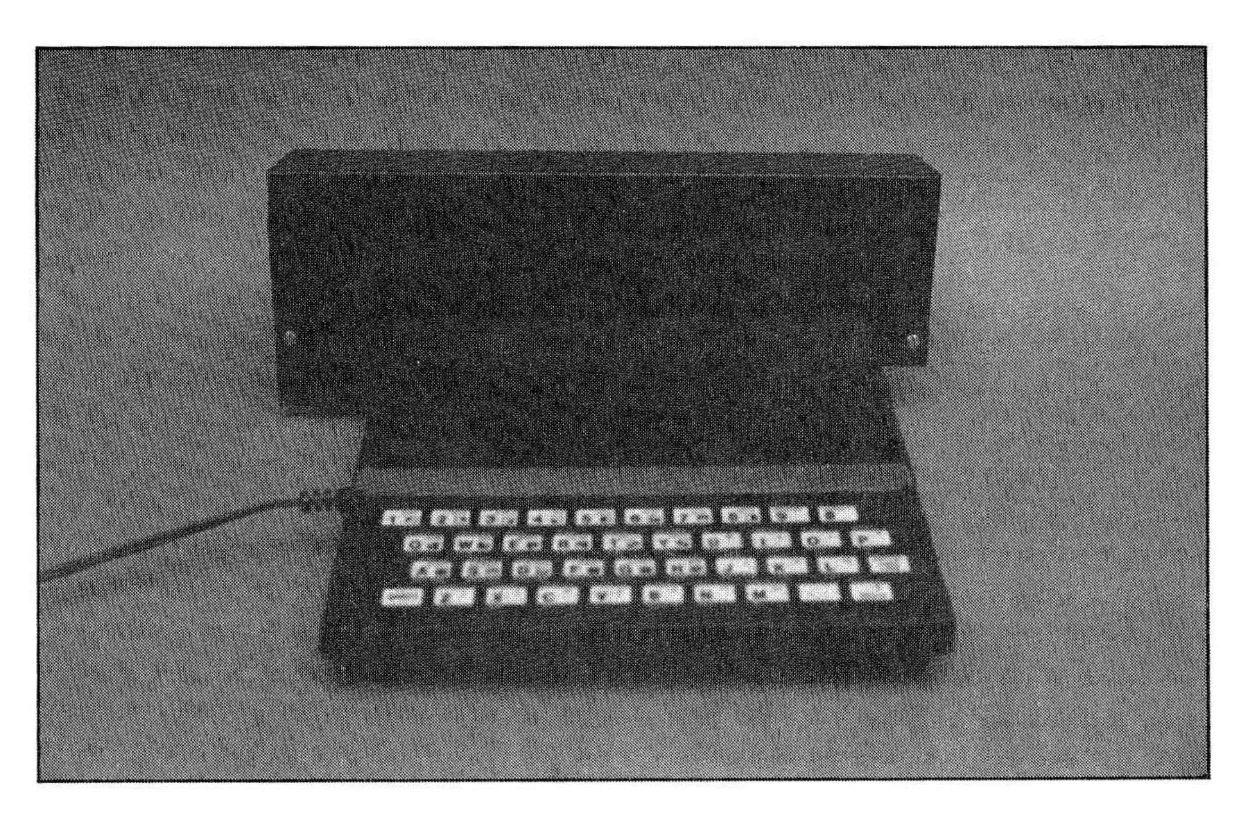
Enfin, on pourra avantageusement prévoir un petit minuteur car il est déconseillé de laisser trop longtemps les EPROMS « sur le feu », et... l'erreur est humaine !

Chapitre 6

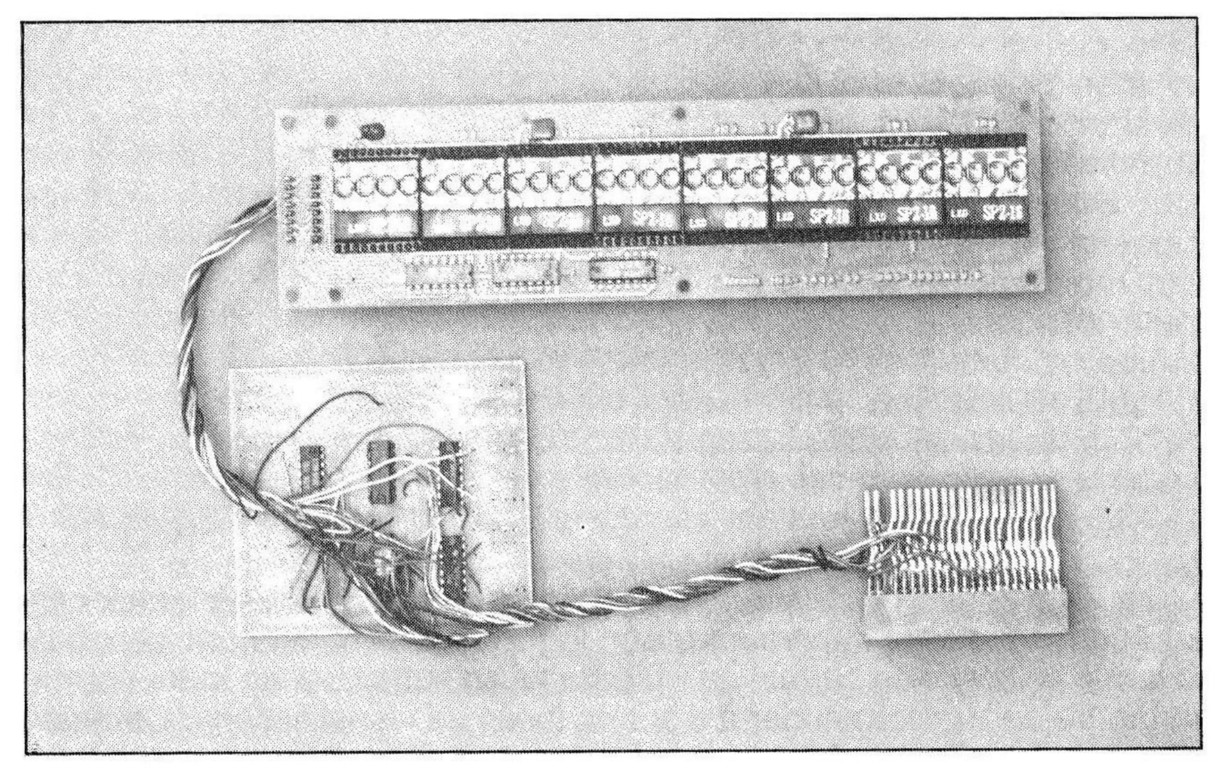
Un afficheur autonome pour le ZX 81



Le ZX 81 est un ordinateur spécialement étudié en vue de fonctionner sur écran TV ou sur imprimante. Son logiciel résident est donc conçu en ce sens, et il est bien certain qu'il faudra lui en substituer un autre pour lui permettre de s'accommoder à d'autres types de visualisations. Comme il n'est guère aisé d'intervenir sur la ROM (par échange ou adjonction), on peut songer à charger des sous-programmes en RAM, qui pourront être appelés à loisir par le programme principal.



Le ZX 81 devient véritablement autonome grâce à cet afficheur permettant de se passer d'écran TV.



Vue d'ensemble des divers éléments de l'afficheur autonome, avant sa « mise en boîte ».

Il ne faudrait cependant pas que les tâches de gestion de l'affichage viennent ralentir exagérément l'exécution du programme principal (on rappelle que le ZX 81 passe les trois quarts du temps à rafraîchir l'écran TV !). Certes, cet écran étant supprimé, on pourra sans arrière-pensée commuter systématiquement la machine en mode rapide (FAST). Malgré tout, il ne saurait être question de confier entièrement au logiciel la gestion d'un afficheur externe.

Il faut donc recourir à des afficheurs dits « intelligents », c'est-à-dire possédant des circuits de mémorisation, de décodage, et d'adressage. Ainsi, la machine n'aura à intervenir *que lors des changements* d'affichage et restera entièrement disponible entre temps. Egalement, le logiciel d'affichage n'occupera de la sorte que quelques lignes de BASIC.

Les afficheurs intelligents se présentent sous la forme de gros circuits intégrés regroupant en général quatre caractères ASCII (soit 64 possibilités de lettres, chiffres et signes divers).

Il serait envisageable d'acheter huit composants de ce type et de graver un circuit imprimé destiné à les recevoir. Cependant, tous calculs faits, il s'avère plus économique d'acquérir directement un module à 32 caractères. On bénéficie de l'effet de série dû à la clientèle industrielle, tout en échappant à la fabrication d'un très complexe circuit double face à trous métallisés.

Ne nous y trompons cependant pas : il s'agit là d'un achat assez onéreux, puisque le module coûte environ autant qu'un téléviseur noir et blanc ! Il faut toutefois comparer des choses comparables et rapprocher le prix d'un ZX 81, muni de cet accessoire, de celui d'autres machines possédant d'origine une ligne d'affichage incorporée. Bien sûr, pour rentabiliser l'investissement, il faut avoir l'usage d'un système entièrement autonome, mais de telles utilisations ne manquent pas !

La réalisation pratique

Après diverses recherches, nous avons porté notre choix sur une barrette LITRONIX IDA 1416-32, qui devrait être disponible lors de la sortie de ce livre (1).

Cet afficheur dispose de caractères de taille identique à ceux apparaissant sur un écran TV de 23 cm, donc tout à fait confortables à la

(1) En cas de problèmes d'approvisionnement, nos lecteurs sont invités à prendre contact avec la société SIEMENS (Tél. (1) 820.61.20), qui distribue les produits LITRONIX en France.

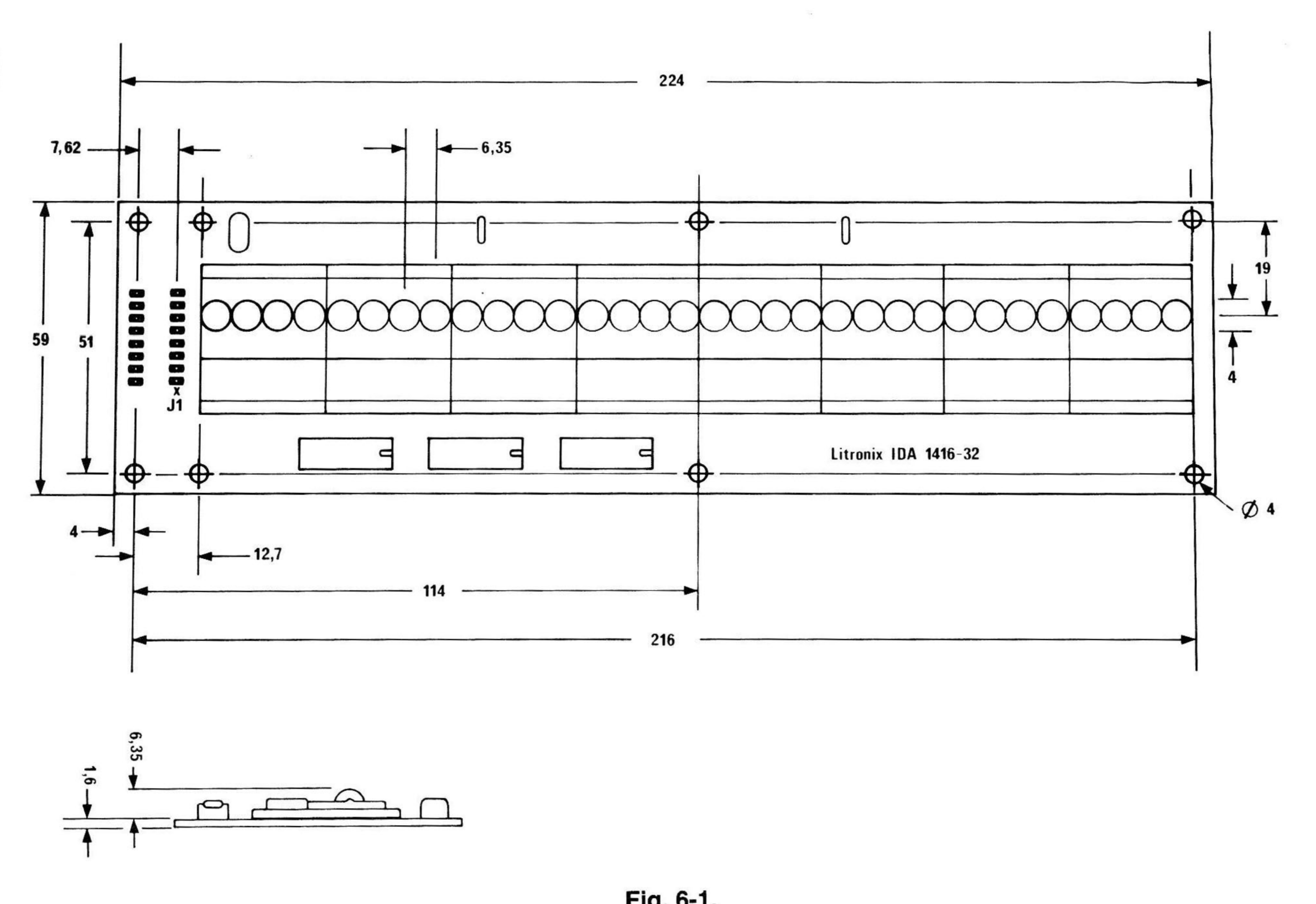
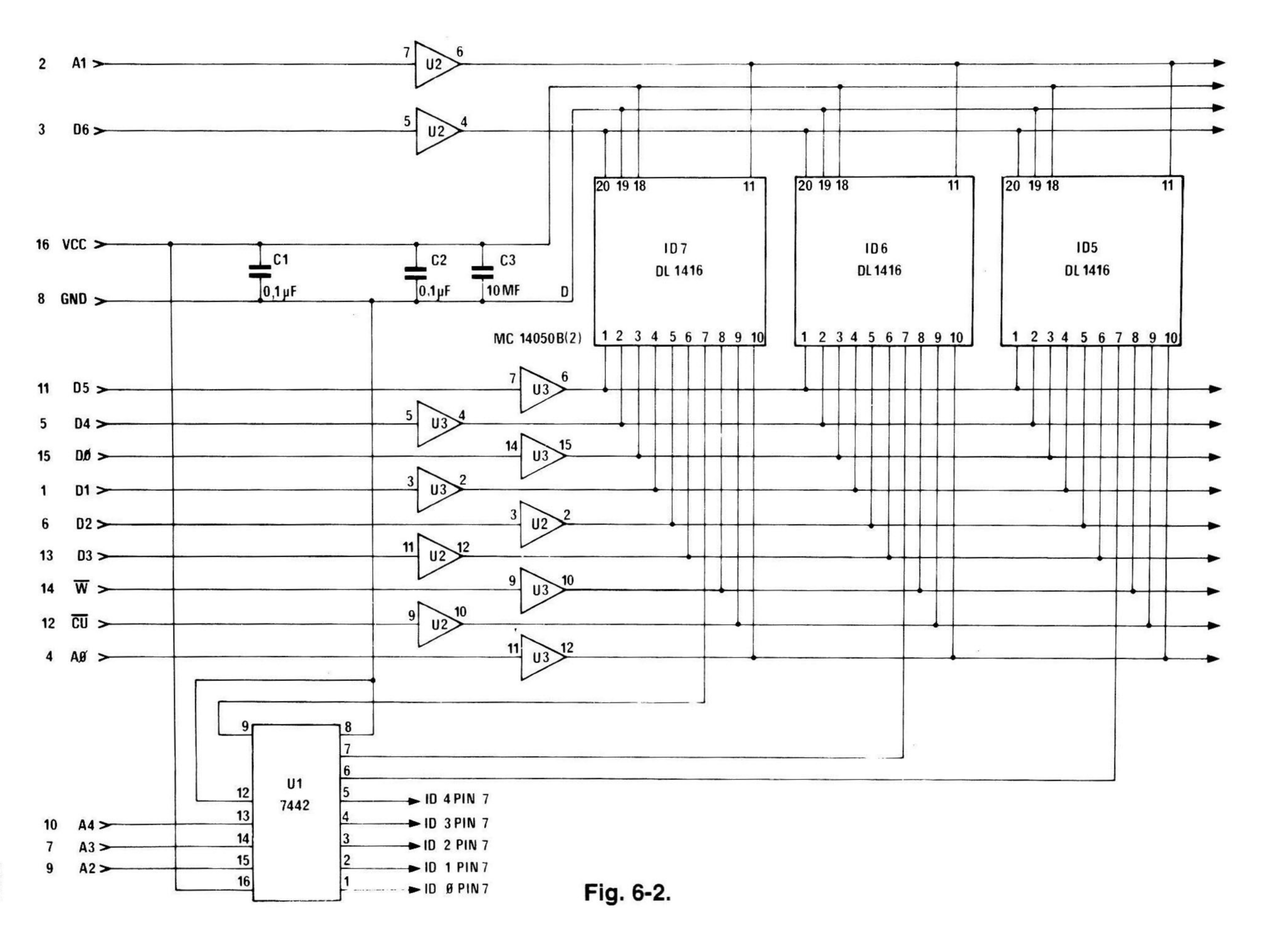

224
7,62
6,35
59
51
J1
19
4
Litronix IDA 1416-32
Ø 4
4
12,7
114
216
6,35
1,6

Fig. 6-1.



2 A1
3 D6
16 VCC
8 GND
C1 0,1µF
C2 0,1µF
C3 10MF
D
MC 14050B(2)
ID7
DL 1416
ID6
DL 1416
ID5
DL 1416
11 D5
5 D4
15 DØ
1 D1
6 D2
13 D3
14 W
12 CU
4 AØ
U1
7442
10 A4
7 A3
9 A2
ID 4 PIN 7
ID 3 PIN 7
ID 2 PIN 7
ID 1 PIN 7
ID Ø PIN 7

Fig. 6-2.

lecture. Il consomme néanmoins suffisamment peu pour pouvoir fonctionner sur le régulateur 5 V incorporé au ZX 81, même en présence de l'imprimante et d'une extension mémoire.

La *figure 6-1* réunit les informations relatives à l'aspect physique de l'afficheur et, en particulier, ses principales dimensions.

Le schéma du module est donné à la *figure 6-2*, ce qui permet de constater la simplicité de son raccordement : cinq lignes d'adresses permettent d'appeler individuellement chacun des 32 caractères, alors que sept lignes de données servent à véhiculer des codes ASCII standards.

A signaler aussi deux broches d'alimentation 5 V et deux entrées de commande, W sur laquelle un niveau bas sert à autoriser l'écriture d'un caractère à l'adresse choisie, et CU, servant à gérer les curseurs.

Le brochage du circuit apparaît à la *figure 6-3*. En la lisant, il importe de noter que l'accès à la carte se fait par l'intermédiaire de seize pastilles disposées comme pour recevoir un circuit intégré DIL. La numérotation des broches obéit d'ailleurs aux règles habituelles en ce domaine (broche n° 1 près du coin inférieur gauche de la rangée d'afficheurs).

1	Ligne de donnée D_1	**9**	Ligne d'adresse A_2
2	Ligne d'adresse A_1	**10**	Ligne d'adresse A_4
3	Ligne de donnée D_6	**11**	Ligne de donnée D_5
4	Ligne d'adresse A_0	**12**	Curseur CU
5	Ligne de donnée D_4	**13**	Ligne de donnée D_3
6	Ligne de donnée D_2	**14**	WR
7	Ligne d'adresse A_3	**15**	Ligne de donnée D_0
8	Masse	**16**	+ 5 V

Fig. 6-3.

Reste donc à organiser le raccordement de ce module au connecteur arrière du ZX 81.

Le chronogramme de l'afficheur étant assez différent de celui du microprocesseur Z 80 A, il est commode de faire transiter les signaux par une *carte d'interface*, telle que notre carte à vingt sorties décrite au chapitre 3.

Cependant, diverses simplifications peuvent être introduites :

- quatorze sorties suffisent (5 adresses, 7 données, 2 commandes) ;
- l'afficheur étant compatible TTL, les transistors d'adaptation de niveau sont superflus, et même nuisibles.

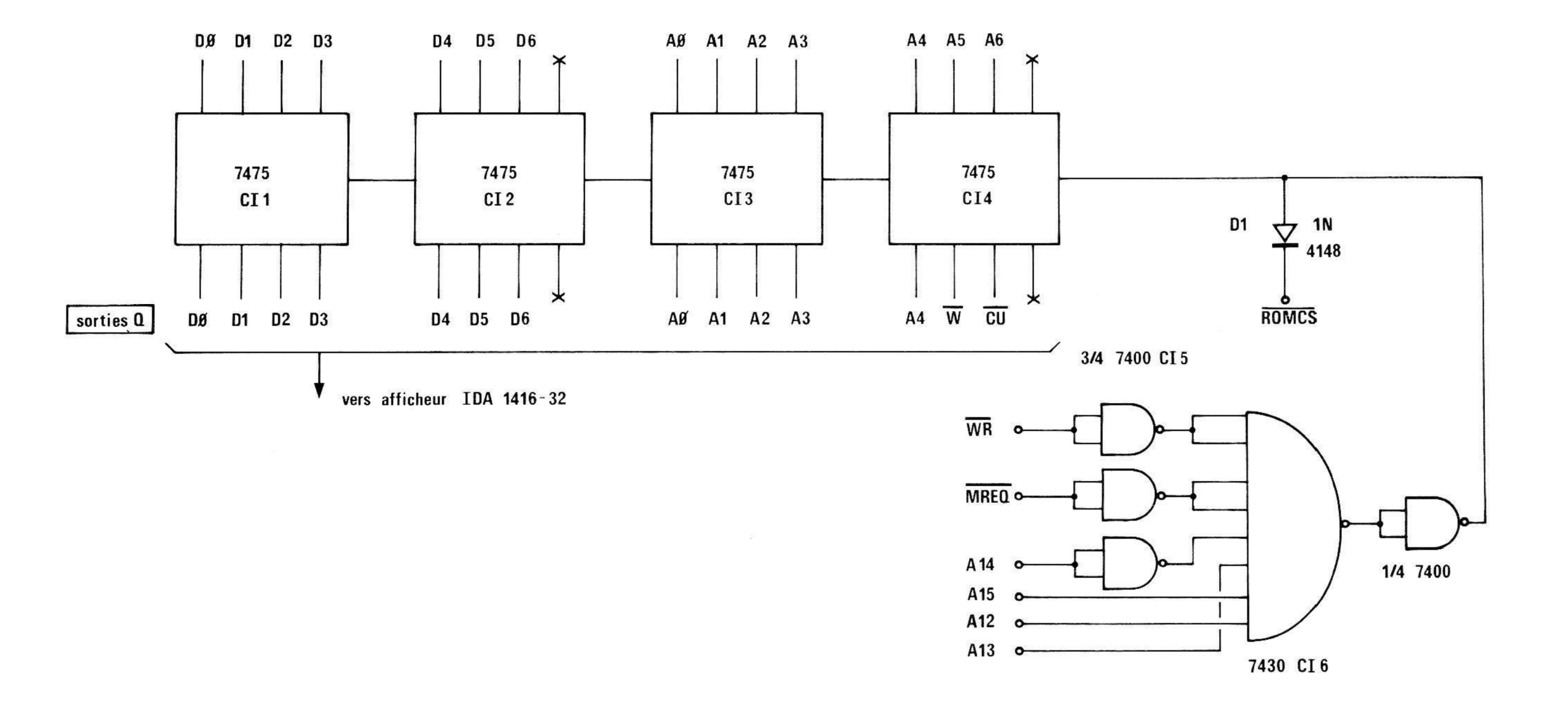

DØ D1 D2 D3
D4 D5 D6
AØ A1 A2 A3
A4 A5 A6
7475
CI 1
7475
CI 2
7475
CI 3
7475
CI 4
D1
1N
4148
ROMCS
sorties Q
DØ D1 D2 D3
D4 D5 D6
AØ A1 A2 A3
A4 W CU
3/4 7400 CI 5
vers afficheur IDA 1416-32
WR
MREQ
A 14
A15
A12
A13
1/4 7400
7430 CI 6

Fig. 6-4.

La *figure 6-4* donne donc le schéma simplifié de la carte. On y remarquera qu'il est fait usage des sorties directes des 7475 (ou 74LS75) alors que notre dernière application utilisait les sorties complémentées.

Au niveau pratique, les conséquences sont les suivantes :

- nous donnons à la *figure 6-5* un tracé universel pour le circuit imprimé : il comporte toutes les liaisons de la version précédemment décrite, plus quatorze pastilles destinées au raccordement de l'afficheur. Certaines liaisons en bord de carte resteront inutilisées lors du câblage selon la *figure 6-6 ;*

- ceux de nos lecteurs qui souhaiteraient réutiliser une carte déjà gravée d'après les plans précédents n'auront qu'à ajouter quatorze pastilles autocollantes en cuivre. De telles pastilles, ainsi que des rubans adhésifs en cuivre de 35 microns, sont disponibles chez les revendeurs

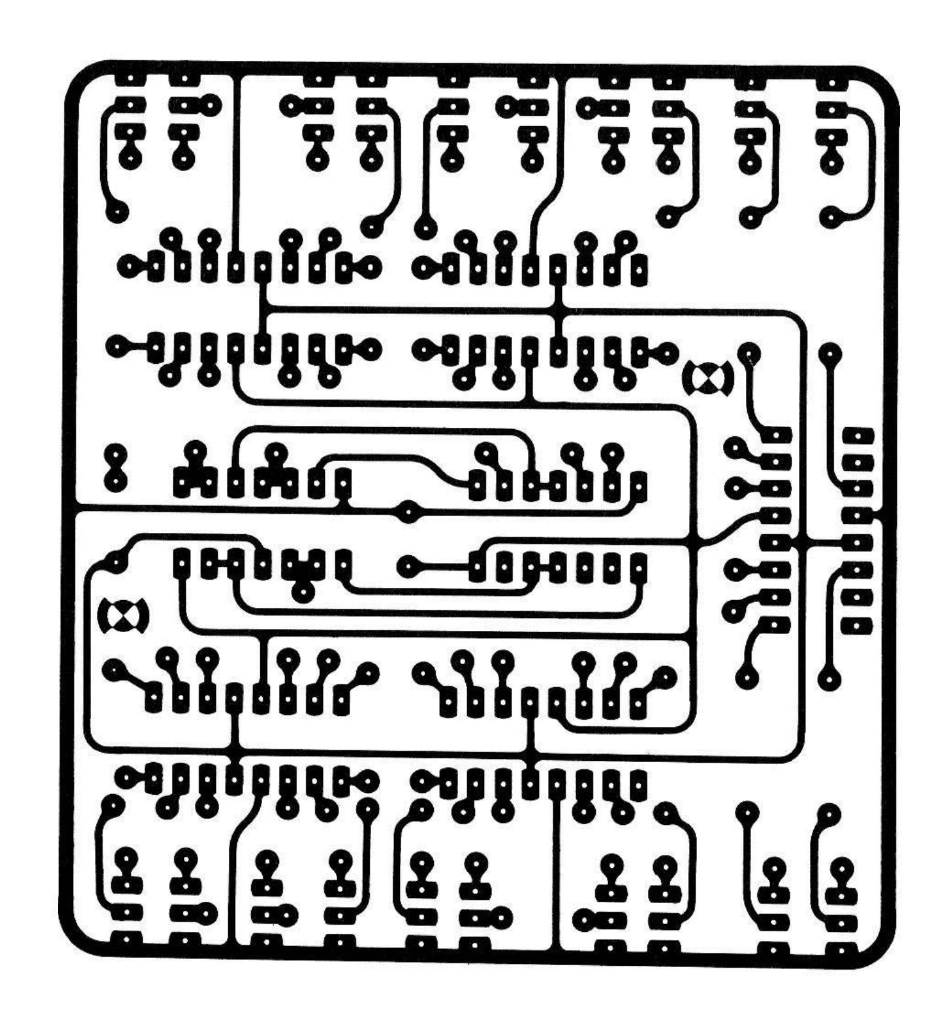


Fig. 6-5.

du *Circuit Imprimé Français* sous la marque EZ *Circuit de Bishop Graphics*. Une fois posés, rien ne distingue de tels éléments du circuit original ;

- insistons sur le fait qu'*il ne faut pas* utiliser les sorties à transistors de la version précédente de cette carte, puisque des collecteurs ouverts ne peuvent en aucun cas commander les entrées CMOS de l'afficheur.

Le raccordement au ZX 81 se fera de façon classique au moyen d'un connecteur gigogne dont la *fig. 6-7* donne le détail de réalisation. Rappelons que la partie femelle doit être obtenue par sciage soigneux d'un connecteur double face au pas de 2,54 mm comptant au moins 25 contacts dont deux seront remplacés par un détrompeur. Dans le cas d'un circuit imprimé de ZX 81 inséré dans un boîtier plus grand abritant déjà d'autres montages, on pourrait se dispenser du connecteur et souder directement les fils provenant de la carte, l'afficheur pouvant même être logé dans une découpe dudit boîtier.

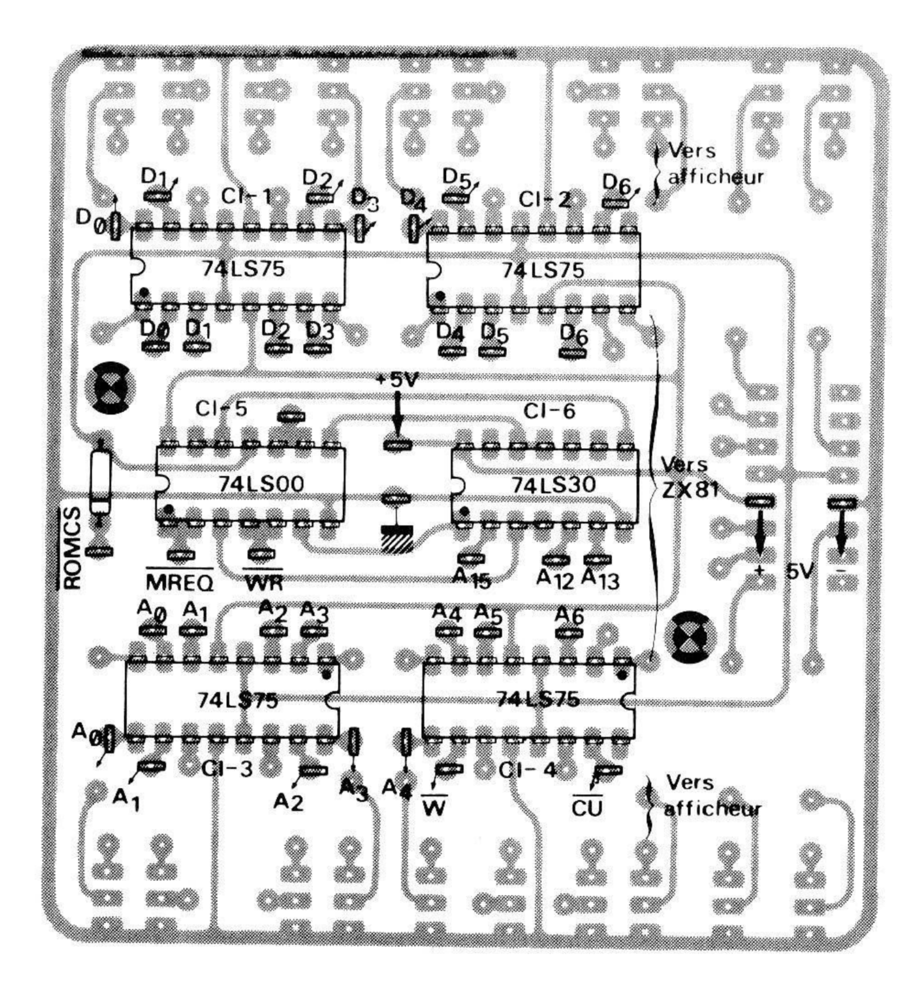


Fig. 6-6.

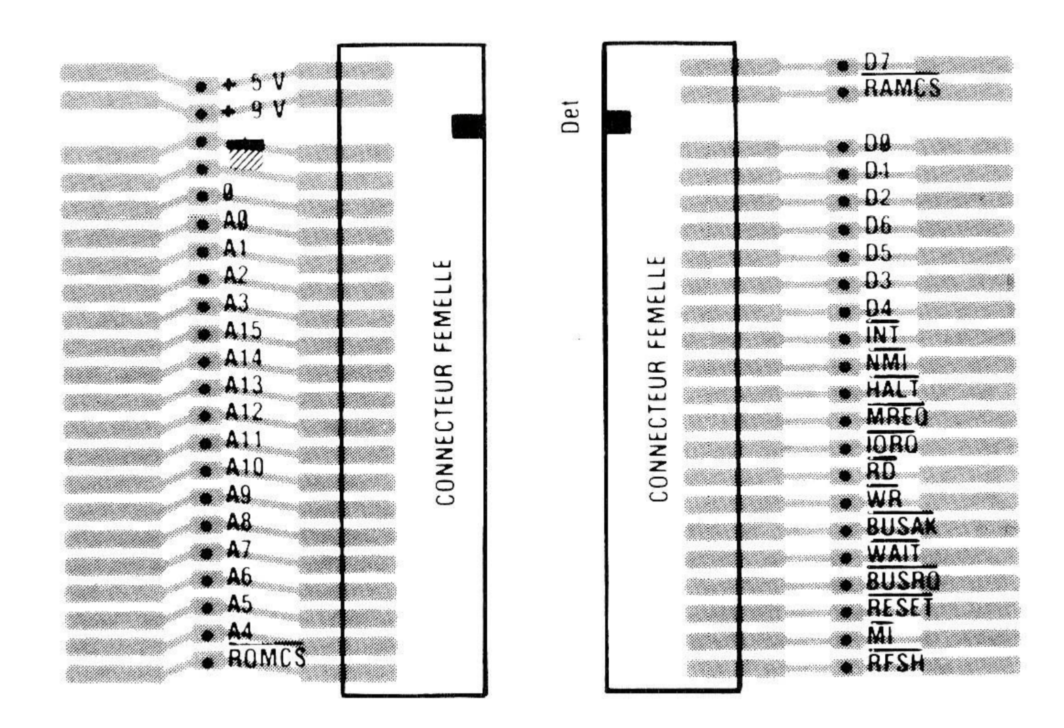
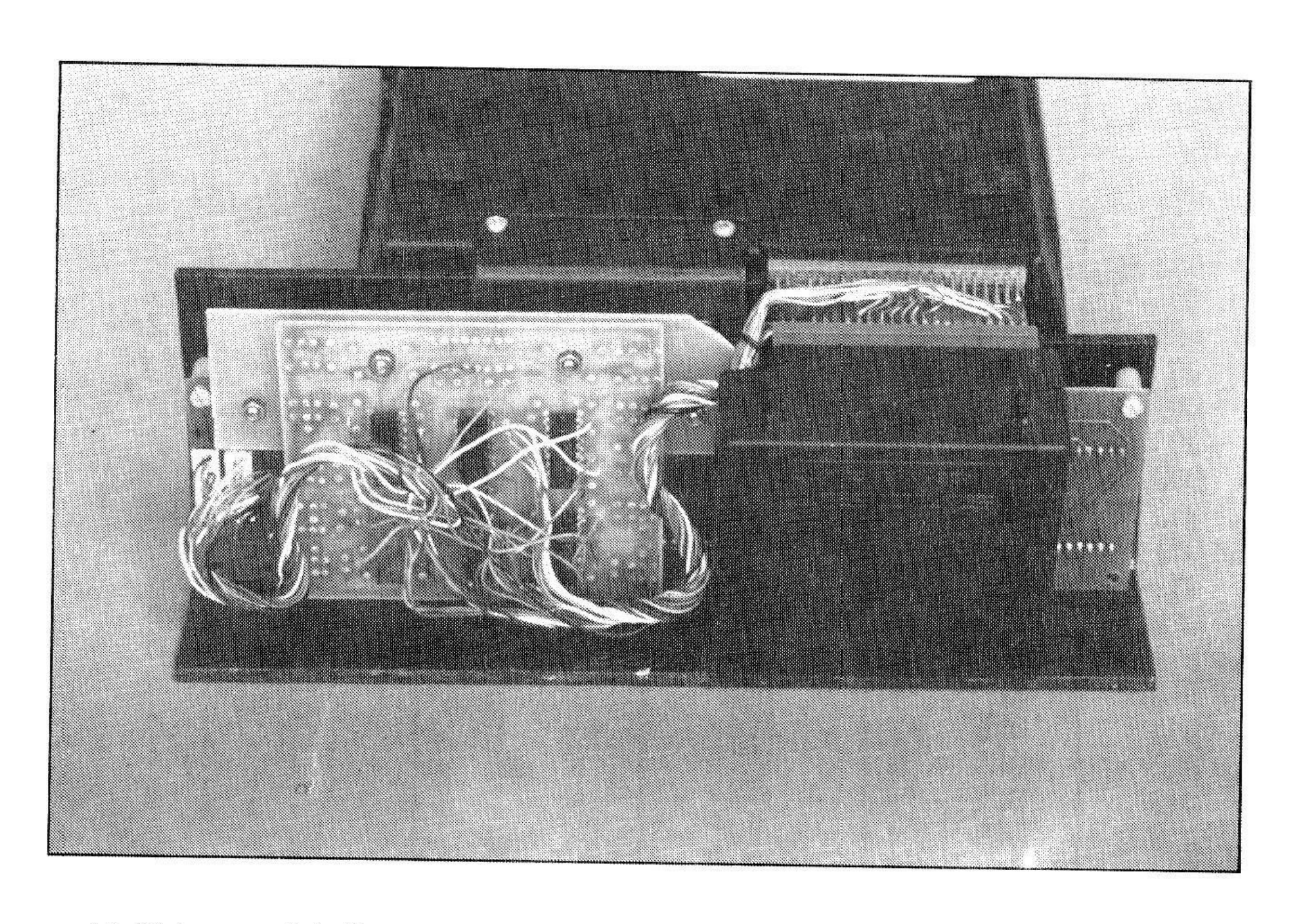


Fig. 6-7.

	D6	D5	D4	D3	0	1	2	3	4	5	6	7
				D0	L	H	L	H	L	H	L	H
				D1	L	L	H	H	L	L	H	H
				D2	L	L	L	L	H	H	H	H
32	L	H	L	L		!	"	#	$	%	&	'
40	L	H	L	H	(	)	*	+	,	-	.	/
48	L	H	H	L	0	1	2	3	4	5	6	7
56	L	H	H	H	8	9	:	;	<	=	>	?
64	H	L	L	L	@	A	B	C	D	E	F	G
72	H	L	L	H	H	I	J	K	L	M	N	O
80	H	L	H	L	P	Q	R	S	T	U	V	W
88	H	L	H	H	X	Y	Z	[	\	]	^	_

Fig. 6-8.



L'afficheur « intelligent » est commandé par une variante de notre carte à vingt sorties.

Le logiciel de commande

Même relié au ZX 81, l'afficheur ne peut être mis en service sans le lancement d'un logiciel approprié. Tout au plus peut-on voir apparaître quelques caractères sans signification lors de la mise sous tension (contenu aléatoire des mémoires des afficheurs intelligents).

L'une des tâches que devra accomplir ce logiciel sera le transcodage Sinclair-ASCII. En effet, la *figure 6-8* reproduit le jeu de caractères dont dispose l'afficheur et qui n'est, ni plus ni moins, que la moitié du code ASCII standard à 7 bits *(fig. 6-9)*.

La *figure 6-10*, quant à elle, reproduit le jeu de caractères du ZX 81, qui peut être décomposé ainsi :

— les 26 lettres de l'alphabet, dont les codes sont décalés de 27 par rapport à l'ASCII (majuscules seulement) ;

— les chiffres de Ø à 9, dont les codes sont décalés de 20 par rapport à l'ASCII ;

Décimal	Caract.	Décimal	Caract.	Décimal	Caract.
000	NUL	043	+		
001	SOH	044	,	086	V
002	STX	045	–	087	W
003	ETX	046	.	088	X
004	EOT	047	/	089	Y
005	ENQ	048	0	090	Z
006	ACK	049	1	091	[
007	BEL	050	2	092	√
008	BS	051	3	093	]
009	HT	052	4	094	Λ
010	LF	053	5	095	←
011	VT	054	6	096	
012	FF	055	7	097	a
013	CR	056	8	098	b
014	SO	057	9	099	c
015	SI	058	:	100	d
016	DLE	059	;	101	e
017	DC1	060	<	102	f
018	DC2	061	=	103	g
019	DC3	062	>	104	h
020	DC4	063	?	105	i
021	NAK	064	@	106	j
022	SYN	065	A	107	k
023	ETB	066	B	108	l
024	CAN	067	C	109	m
025	EM	068	D	110	n
026	SUB	069	E	111	o
027	ESCAPE	070	F	112	p
028	FS	071	G	113	q
029	GS	072	H	114	r
030	RS	073	I	115	s
031	US	074	J	116	t
032	SPACE	075	K	117	u
033	!	076	L	118	v
034	''	077	M	119	w
035	#	078	N	120	x
036	■	079	O	121	y
037	%	080	P	122	z
038	&	081	Q	123	{
039	'	082	R	124	/
040	(	083	S	125	}
041	)	084	T	126	
042	*	085	U	127	DEL

caractères non disponibles sur l'afficheur

caractères disponibles sur l'afficheur

caractères non disponibles sur l'afficheur

Fig. 6-9. LF = interligne (line feed) CR = retour chariot
FF = présentation de feuille (form feed) DEL = effacement sur le télétype

0	:	1	: ▘
2	: ▝	3	: ▀
4	: ▖	5	: ▌
6	: ▞	7	: ▛
8	: ▒	9	: ▒
10	: ▒	11	: "
12	: £	13	: $
14	: :	15	: ?
16	: (	17	:)
18	: >	19	: <
20	: =	21	: +
22	: -	23	: *
24	: /	25	: ;
26	: ,	27	: .
28	: 0	29	: 1
30	: 2	31	: 3
32	: 4	33	: 5
34	: 6	35	: 7
36	: 8	37	: 9
38	: A	39	: B
40	: C	41	: D
42	: E	43	: F
44	: G	45	: H
46	: I	47	: J
48	: K	49	: L
50	: M	51	: N
52	: O	53	: P
54	: Q	55	: R
56	: S	57	: T
58	: U	59	: V
60	: W	61	: X
62	: Y	63	: Z
64	: RND	65	: INKEY$
66	: PI	67	: ?
68	: ?	69	: ?
70	: ?	71	: ?
72	: ?	73	: ?
74	: ?	75	: ?
76	: ?	77	: ?
78	: ?	79	: ?
80	: ?	81	: ?
82	: ?	83	: ?
84	: ?	85	: ?
86	: ?	87	: ?
88	: ?	89	: ?
90	: ?	91	: ?
92	: ?	93	: ?
94	: ?	95	: ?
96	: ?	97	: ?
98	: ?	99	: ?
100	: ?	101	: ?
102	: ?	103	: ?
104	: ?	105	: ?
106	: ?	107	: ?
108	: ?	109	: ?

Fig. 6-10.
suite au verso

110	:	?	111	:	?
112	:	?	113	:	?
114	:	?	115	:	?
116	:	?	117	:	?
118	:	?	119	:	?
120	:	?	121	:	?
122	:	?	123	:	?
124	:	?	125	:	?
126	:	?	127	:	?
128	:	[illegible]	129	:	[illegible]
130	:	[illegible]	131	:	[illegible]
132	:	[illegible]	133	:	[illegible]
134	:	[illegible]	135	:	[illegible]
136	:	[illegible]	137	:	[illegible]
138	:	[illegible]	139	:	[illegible]
140	:	[illegible]	141	:	[illegible]
142	:	[illegible]	143	:	[illegible]
144	:	[illegible]	145	:	[illegible]
146	:	[illegible]	147	:	[illegible]
148	:	[illegible]	149	:	[illegible]
150	:	[illegible]	151	:	[illegible]
152	:	[illegible]	153	:	[illegible]
154	:	[illegible]	155	:	[illegible]
156	:	[illegible]	157	:	[illegible]
158	:	[illegible]	159	:	[illegible]
160	:	[illegible]	161	:	[illegible]
162	:	[illegible]	163	:	[illegible]
164	:	[illegible]	165	:	[illegible]
166	:	[illegible]	167	:	[illegible]
168	:	[illegible]	169	:	[illegible]
170	:	[illegible]	171	:	[illegible]
172	:	[illegible]	173	:	[illegible]
174	:	[illegible]	175	:	[illegible]
176	:	[illegible]	177	:	[illegible]
178	:	[illegible]	179	:	[illegible]
180	:	[illegible]	181	:	[illegible]
182	:	[illegible]	183	:	[illegible]
184	:	[illegible]	185	:	[illegible]
186	:	[illegible]	187	:	[illegible]
188	:	[illegible]	189	:	[illegible]
190	:	[illegible]	191	:	[illegible]
192	:	"	193	:	AT
194	:	TAB	195	:	?
196	:	CODE	197	:	VAL
198	:	LEN	199	:	SIN
200	:	COS	201	:	TAN
202	:	ASN	203	:	ACS
204	:	ATN	205	:	LN
206	:	EXP	207	:	INT
208	:	SQR	209	:	SGN
210	:	ABS	211	:	PEEK
212	:	USR	213	:	STR$
214	:	CHR$	215	:	NOT
216	:	**	217	:	OR
218	:	AND	219	:	<=
220	:	>=	221	:	<>

222	:	THEN	223	:	TO
224	:	STEP	225	:	LPRINT
226	:	LLIST	227	:	STOP
228	:	SLOW	229	:	FAST
230	:	NEW	231	:	SCROLL
232	:	CONT	233	:	DIM
234	:	REM	235	:	FOR
236	:	GOTO	237	:	GOSUB
238	:	INPUT	239	:	LOAD
240	:	LIST	241	:	LET
242	:	PAUSE	243	:	NEXT
244	:	POKE	245	:	PRINT
246	:	PLOT	247	:	RUN
248	:	SAVE	249	:	RAND
250	:	IF	251	:	CLS
252	:	UNPLOT	253	:	CLEAR
254	:	RETURN	255	:	COPY

Fig. 6-10.

— les signes usuels (ponctuation, mathématiques), pour lesquels la correspondance avec l'ASCII est à examiner cas par cas ;
— les caractères graphiques, intraduisibles en ASCII ;
— les codes de contrôle, représentés par des ? et qui n'ont en aucun cas à être affichés ;
— les mots-clé qui, à l'affichage, peuvent être reconstitués par assemblage de caractères classiques ;
— les caractères inverses qui ne diffèrent de leurs modèles que par une translation de 128.

Un transcodage intégral n'est donc pas possible, le principal obstacle se situant au niveau des caractères graphiques. Toutefois, l'intérêt de ces caractères n'apparaît que sur un écran TV entier, et non sur une simple ligne de trente-deux caractères.

Selon l'encombrement mémoire admissible pour le logiciel d'affichage, on peut prévoir un transcodage plus ou moins complet, dont les *figure 6-11 et 6-12* donnent deux exemples extrêmes.

Le programme de la *figure 6-11* ne traite que les lettres et chiffres, ce qui, soit dit en passant, se révèle suffisant pour bon nombre d'applications. Son principal intérêt est de fonctionner sans extension mémoire (avec 1K RAM), tout en laissant un peu de place pour un logiciel utilisateur gérant, par exemple, des circuits d'entrée-sortie.

Comme il ne saurait être question d'utiliser des PRINT, il a été décidé de loger le texte à afficher dans une simple chaîne nommée A$, puis

d'appeler un sous-programme. Sur notre exemple, on appelle même deux routines successivement, GOSUB 9995 servant à effacer rapidement la ligne précédente, alors que GOSUB 8000 procède à l'écriture proprement dite.

Avec la *figure 6-12*, nous disposons d'un transcodage à peu près aussi complet que possible, mais il est nécessaire de recourir à une extension de RAM. Notons que l'afficheur lui-même affecte automatiquement des espaces aux codes qui, après transcodage, ne correspondent à aucune des combinaisons de la *figure 6-8*.

Ces deux programmes fonctionnent bien sûr en modes rapide et lent, mais il est évident que le mode FAST s'impose en l'absence d'écran TV.

Remarquons que rien n'empêche d'utiliser simultanément l'écran TV et l'afficheur, en toute indépendance, ce qui peut ouvrir certaines perspectives de configurations multi-utilisateurs, dans le domaine des jeux ou d'applications beaucoup plus sérieuses...

```
   1 LET A$="AFFICHEUR POUR SINC
LAIR ZX 81"
   2 GOSUB 9995
   3 GOSUB 8000
   4 PAUSE 200
   5 LET A$="32 CARACTERES ALPHA
NUMERIQUES"
   6 GOSUB 9995
   7 GOSUB 8000
   8 PAUSE 200
   9 RUN
7000 REM COPYRIGHT 1983
8000 IF LEN A$>=32 THEN GOTO 804
0
8010 LET A$=A$+CHR$ 0
8020 GOTO 8000
8040 FOR F=31 TO 0 STEP -1
8050 LET D=CODE A$(32-F)
8060 IF D>=38 AND D<=63 THEN LET
 D=D+27
8070 IF D>=28 AND D<=37 THEN LET
 D=D+20
8080 POKE 45120+F,D
8090 POKE 45056+F,D
8100 POKE 45152+F,D
8110 NEXT F
8120 GOSUB 9991
8130 RETURN
9991 FOR F=0 TO 31
9992 POKE 45056+F,0
9993 NEXT F
9994 RETURN
```

```
9995 FOR F=0 TO 31
9996 POKE 45120+F,0
9997 POKE 45152+F,0
9998 NEXT F
9999 RETURN
```

Fig. 6-11.

```
   1 LET A$="Y COMPRIS LES PRINC
IPAUX SIGNES"
   2 GOSUB 9995
   3 GOSUB 8000
   4 PAUSE 200
   5 LET A$="OUI,VRAIMENT ? ALOR
S 2*4=8..."
   6 GOSUB 9995
   7 GOSUB 8000
   8 PAUSE 200
   9 RUN
8000 IF LEN A$>=32 THEN GOTO 804
0
8010 LET A$=A$+CHR$ 0
8020 GOTO 8000
8040 FOR F=31 TO 0 STEP -1
8050 LET D=CODE A$(32-F)
8055 IF D>=128 THEN LET D=D-128
8060 IF D>=38 AND D<=63 THEN LET
 D=D+27
8070 IF D>=28 AND D<=37 THEN LET
 D=D+20
8080 IF D>=11 AND D<=13 THEN LET
 D=D+23
8090 IF D=14 OR D=18 THEN LET D=
D+44
8100 IF D=15 THEN LET D=63
8110 IF D=16 OR D=17 THEN LET D=
D+24
8120 IF D=19 OR D=20 THEN LET D=
D+41
8130 IF D=21 THEN LET D=43
8140 IF D=22 OR D=24 THEN LET D=
D+23
8150 IF D=23 OR D=27 THEN LET D=
D+19
8160 IF D=25 THEN LET D=59
8170 IF D=26 THEN LET D=44
8180 POKE 45120+F,D
8190 POKE 45056+F,D
8200 POKE 45152+F,D
8210 NEXT F
8220 GOSUB 9991
8230 RETURN
9991 FOR F=0 TO 31
9992 POKE 45056+F,0
```

Fig. 6-12.
suite au verso

```
9993 NEXT F
9994 RETURN
9995 FOR F=0 TO 31
9996 POKE 45120+F,0
9997 POKE 45152+F,0
9998 NEXT F
9999 RETURN
```

Fig. 6-12.

Quelques applications

Il est bien certain qu'un ZX 81 équipé de cet afficheur ne peut donner sa pleine mesure : en effet, il perd pratiquement toutes ses possibilités graphiques. Cependant, ce n'est pas pour jouer aux astéroïdes que l'on a besoin d'un système informatique portatif ! La puissance de calcul et de traitement de texte du ZX reste intacte avec une seule ligne d'affichage alphanumérique. Dès lors, on dispose en tous lieux de l'outil informatique pour la résolution de problèmes qui peuvent atteindre un haut degré de complexité.

Dans le domaine de la « robotique domestique », qui nous intéresse plus particulièrement ici, cet afficheur sera surtout utile pour lier un dialogue simple avec un utilisateur non informaticien, ne disposant que de quelques boutons-poussoir pour faire entendre sa « voix ».

On peut ainsi imaginer la réalisation de « portiers » très sophistiqués, donnant des consignes ou des informations personnalisées selon le visiteur qui se sera, par exemple, identifié au moyen d'un code.

Chapitre 7

Une carte sonore adaptable au ZX 81



Il existe deux principales méthodes permettant de faire générer des signaux audibles à un micro-ordinateur ou à un microprocesseur :

La première, qui présente l'avantage certain de n'exiger aucun complément matériel sur la plupart des machines, consiste à « manipuler » très rapidement l'état logique d'un port de sortie entre deux états, au moyen de routines en langage machine. Toute médaille ayant son revers, il faut remarquer que ce procédé est très lourd à mettre en œuvre, que les

sonorités pouvant être créées ne sont en général pas très élaborées et, surtout, que l'unité centrale de la machine est complètement monopolisée tant que dure l'émission du son. On retrouve ici un inconvénient analogue à celui du ZX 81 qui, en mode lent, ne peut effectuer, que pendant de courtes périodes, des tâches autres que la gestion de l'écran TV.

Pour parvenir à une efficacité satisfaisante, il est à peu près indispensable de limiter le rôle de l'unité centrale à l'élaboration d'ordres immédiatement transmis à un « périphérique » spécialisé, capable de les exécuter en disposant de tout le temps nécessaire, alors que l'unité centrale vaque à d'autres occupations.

C'est précisément de la réalisation d'un tel périphérique que va traiter la suite de ce chapitre.

Réalisation pratique

Les circuits générateurs de sons

Depuis quelque temps, des circuits intégrés spécialisés dans la génération de sons complexes sont apparus sur le marché. Pour la plupart, la « programmation » du son à réaliser est obtenue par commutation de résistances et/ou de condensateurs. Il en résulte que la recherche d'effets intéressants passe nécessairement par d'assez lourdes manipulations de connexions.

Le AY-3-8910 de Général Instrument est conçu selon un principe radicalement différent : ce microcircuit possède en effet un jeu de lignes d'entrée-sortie en parallèle, sur lesquelles il suffit d'envoyer des mots binaires décrivant dans un code bien défini toutes les caractéristiques du son à synthétiser.

Dès lors, l'adaptation de ce composant à un micro-ordinateur passera par la résolution de problèmes de deux ordres :

— mise au point de circuits d'interface (matériel),
— écriture des programmes appropriés (logiciel).

C'est bien sûr cette dernière étape qui permettra d'exploiter plus ou moins à fond les possibilités très vastes du système, l'imagination de l'utilisateur étant la principale limite !

Interfaçage du AY-3-8910

Schématiquement, le AY-3-8910 est un générateur de sons à trois canaux, chacun d'entre eux pouvant diffuser, sur une sortie distincte, une

tonalité de fréquence réglable, un bruit « de souffle » de sonorité ajustable ou les deux à la fois. Le volume sonore de chaque voie peut être également programmé sur une très large plage ou bien commandé de façon automatique par un *générateur d'enveloppes* fixant les modalités de croissance et de décroissance du son selon des données préprogrammées.

En réalité, les trois canaux ne sont pas entièrement indépendants car, si la hauteur des notes et le volume « manuel » de chacun peuvent être fixés séparément, les générateurs d'enveloppe et de bruit sont communs aux trois voies. Il ne sera pas possible, notamment, d'affecter une forme d'enveloppe à une tonalité et, simultanément, une autre à du bruit blanc.

Les sorties (1 V crête maximum) des trois voies peuvent être utilisées séparément (stéréophonie à deux ou trois canaux), ou mélangées pour constituer un signal monophonique complexe.

Le circuit s'alimente en + 5 V, consomme environ 75 mA et a besoin d'un signal d'horloge de fréquence comprise entre 1 et 2 MHz. La connaissance précise de cette fréquence est primordiale car c'est à partir de celle-ci que sont obtenus, par division, tous les sons pouvant être synthétisés. L'entrée des ordres s'effectue au moyen d'un bus à dix lignes « trois états », associé à trois lignes « de commande » permettant de sélectionner l'état de ce bus.

Il s'agit en effet d'un bus *multiplexé*, véhiculant tour à tour des adresses et des données, tant en entrée qu'en sortie (pour des applications spéciales).

Les données représentent les ordres codés décrivant les sons à synthétiser, alors que les adresses sont celles de seize « registres » internes entre lesquels ces données doivent être soigneusement ventilées. En fait, la génération de sons n'utilise que quatorze registres, les deux autres pouvant être affectés à deux ports d'entrée-sortie parallèles dont nous n'avons pas l'usage dans le cadre de la synthèse de sons, mais qui restent éventuellement disponibles.

Le tableau de la *figure 7-1* rassemble les informations concernant l'ensemble de ces registres, lesquelles feront l'objet des développements indispensables au fur et à mesure de leur utilisation pratique.

Le cas particulier du Z 80

Le bus du AY-3-8910 a visiblement été étudié en vue d'un raccordement à un microprocesseur à bus multiplexé, tel que le 6800, le 8080, ou les 1600, 1610, 1650 de Général Instrument comme il se doit... Il est également prévu d'utiliser une horloge de 1,78977 MHz.

REGISTER \ BIT		B7	B6	B5	B4	B3	B2	B1	B0
R0	Channel A Tone Period	8-BIT Fine Tune A							
R1						4-BIT Coarse Tune A			
R2	Channel B Tone Period	8-BIT Fine Tune B							
R3						4-BIT Coarse Tune B			
R4	Channel C Tone Period	8-BIT Fine Tune C							
R5						4-BIT Coarse Tune C			
R6	Noise Period				5-BIT Period Control				
R7	$\overline{\text{Enable}}$	$\overline{\text{IN}}$/OUT		$\overline{\text{Noise}}$			$\overline{\text{Tone}}$		
		IOB	IOA	C	B	A	C	B	A
R8	Channel A Amplitude				M	L3	L2	L1	L0
R9	Channel B Amplitude				M	L3	L2	L1	L0
R10	Channel C Amplitude				M	L3	L2	L1	L0
R11	Envelope Period	8-BIT Fine Tune E							
R12		8-BIT Coarse Tune E							
R13	Envelope Shape/Cycle					CONT	ATT	ALT	HOLD
R14	I/O Port A Data Store	8-BIT PARALLEL I/O on Port A							
R15	I/O Port B Data Store	8-BIT PARALLEL I/O Port B							

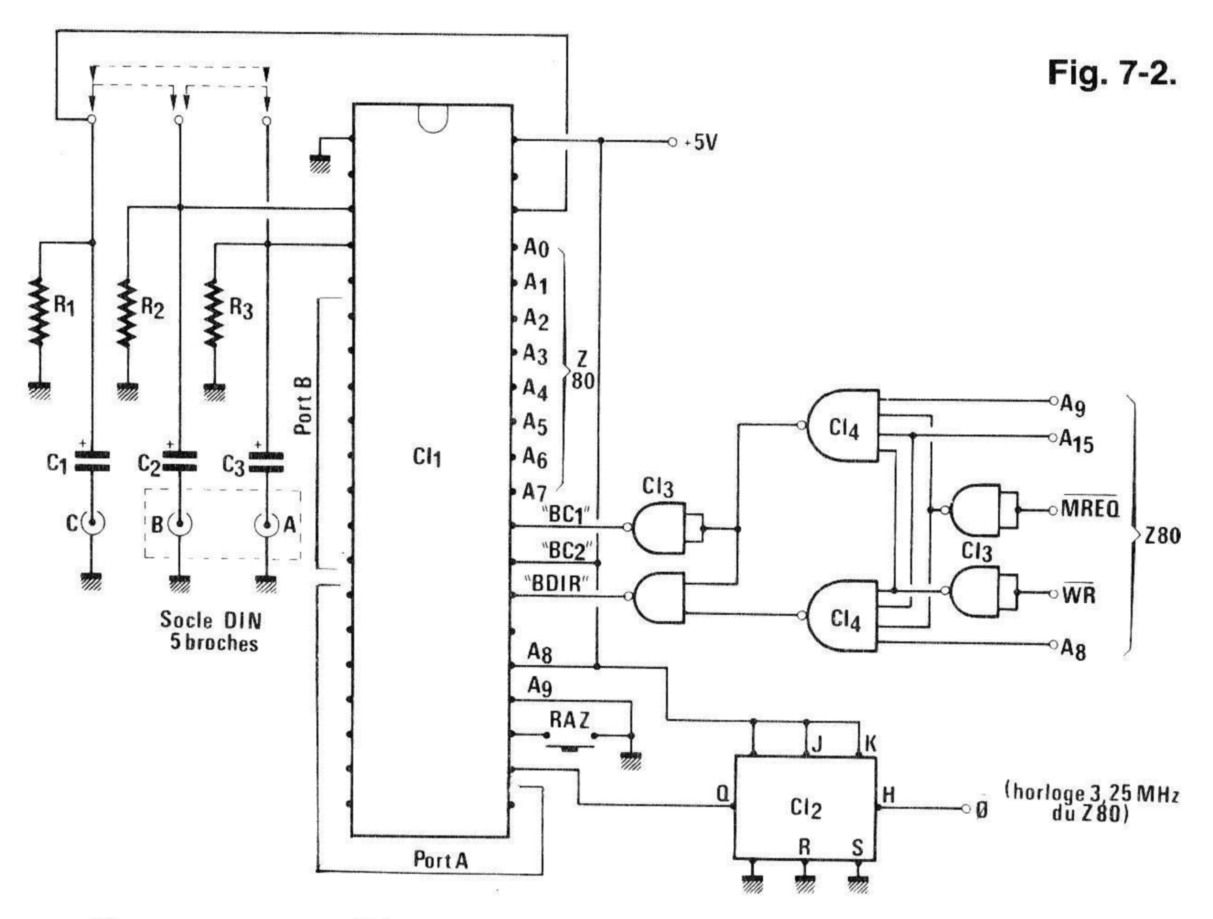
Fig. 7-1.

Pourtant, beaucoup d'ordinateurs individuels, et en particulier le ZX 81, font appel à un microprocesseur Z 80 qui, lui, utilise des bus séparés pour les adresses et les données. De plus, l'horloge du ZX 81 fonctionne sous 3,25 MHz.

Il nous a donc fallu recourir à différents artifices pour aboutir au schéma de la *figure 7-2*, directement compatible avec le ZX 81 :

- division par deux de la fréquence d'horloge, qui passe ainsi à 1,625 MHz ;
- utilisation du seul bus d'adresses, sur lequel un logiciel approprié enverra également les données ;
- élaboration des signaux de commande du AY-3-8910 à partir des signaux disponibles sur le Z 80, à savoir $\overline{MREQ}$, $\overline{WR}$, et certaines lignes d'adresse.

A côté de cela, le ZX 81 pose un problème de taille : son microprocesseur est un Z 80 A, version « rapide » du Z 80 qui délivre signaux de commande, adresses et données pendant des durées inférieures de moitié aux durées minimales nécessaires au générateur de



sons. Dans ces conditions, nous avons pu faire la triste expérience suivante : le synthétiseur refusait quatre ordres sur cinq, et parfois plus !

Comme il ne saurait être question de « latcher » une seconde fois, par un montage externe, tous ces signaux, nous avons pris le parti de « matraquer » le synthétiseur en lui répétant chaque ordre deux cent cinquante-quatre fois !

Ce procédé donne toutes les garanties souhaitables de prise en compte des ordres, mais reste trop lourd pour être mis en œuvre en BASIC. Une courte routine en langage machine désassemblée à la *figure 7-3* peut par contre s'acquitter de cette tâche en une fraction de seconde.

16514	LD B,254	006	254	
16516	LD A,Ø	062	000	
16518	LD(NN),A	050	000	130
16521	LD(NN),A	050	000	129
16524	DEC B	005		
16525	RET Z	200		
16526	JP 16518	195	134	064

Fig. 7-3.

N.B. L'octet 130 en 16520 correspond à la mise à 1 des bits 9 et 15 du bus d'adresses.
L'octet placé en 16519 déterminera les huit bits de poids faible de ce même bus. (AØ à A7).
L'octet 129 en 16523 correspond à la mise à 1 des bits 8 et 15 du bus d'adresses.
L'octet placé en 16522 déterminera les huit bits de poids faible de ce même bus. (AØ à A7).

Cette routine se charge, en plus, de l'indispensable « liaison » entre le BASIC et les circuits d'interface, simplifiant ainsi considérablement la tâche du programmeur.

La mise à 1 du bit 15 du bus d'adresses signifie que le synthétiseur est « adressé » dans l'espace mémoire situé au-dessus d'un éventuel module 16 K (possesseurs de blocs de capacité supérieure s'abstenir). Les bits 8 et 9 de ce même bus indiquent si celui-ci véhicule une adresse de registre (A9 = 1) ou une donnée (A8 = 1).

Les bits AØ à A3 véhiculent les adresses de registres, alors que A0 à A7 servent au transfert de données.

Ces choix ont été effectués de façon à simplifier autant qu'il était possible le circuit d'interface, limité à six portes NAND en technologie CMOS. Notons que durant toutes les opérations de communication entre le synthétiseur et le Z 80, le contenu du bus de données de ce dernier est totalement indifférent (nous lui avons affecté arbitrairement la valeur zéro).

Réalisation pratique du synthétiseur

Les circuits du synthétiseur proprement dit sont logés sur un circuit imprimé simple face dont la *figure 7-4* donne le tracé des pistes. Le plan d'implantation de la *figure 7-5* mentionne deux straps que l'on veillera à ne pas omettre. Le raccordement au ZX 81 s'effectue au moyen d'un connecteur indépendant associé à un petit circuit double face décrit à la *figure 7-6*.

Cette disposition a été retenue pour plusieurs raisons :

- éviter le recours à un complexe circuit double face pour le générateur proprement dit ;
- échapper aux empilages branlants sur le connecteur arrière du ZX 81, surtout si le module 16 K et l'imprimante sont utilisés en même temps ;

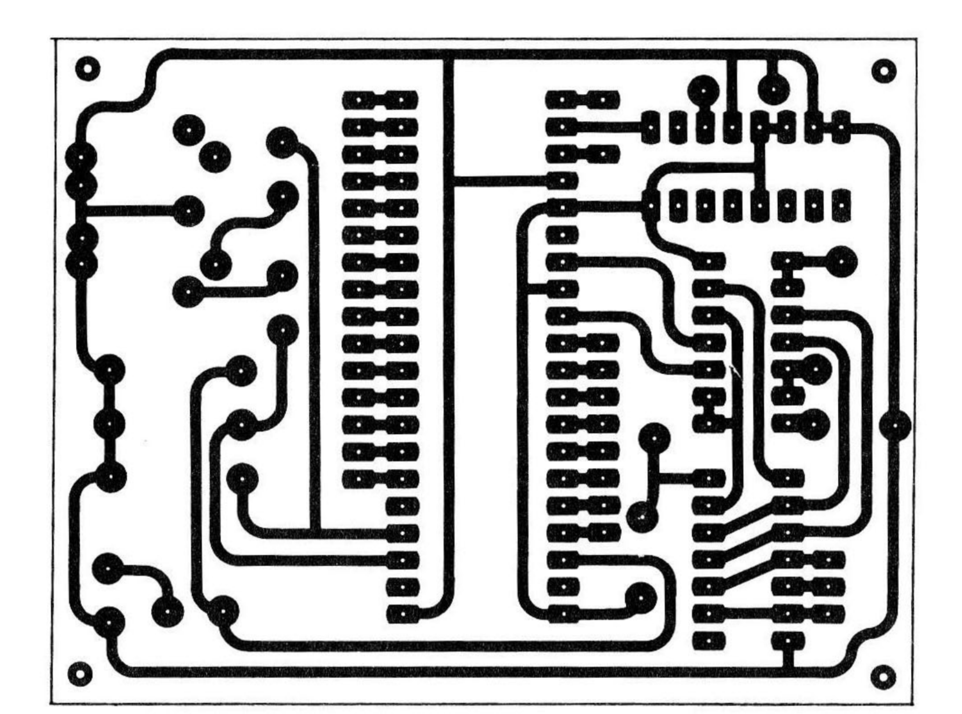


Fig. 7-4.

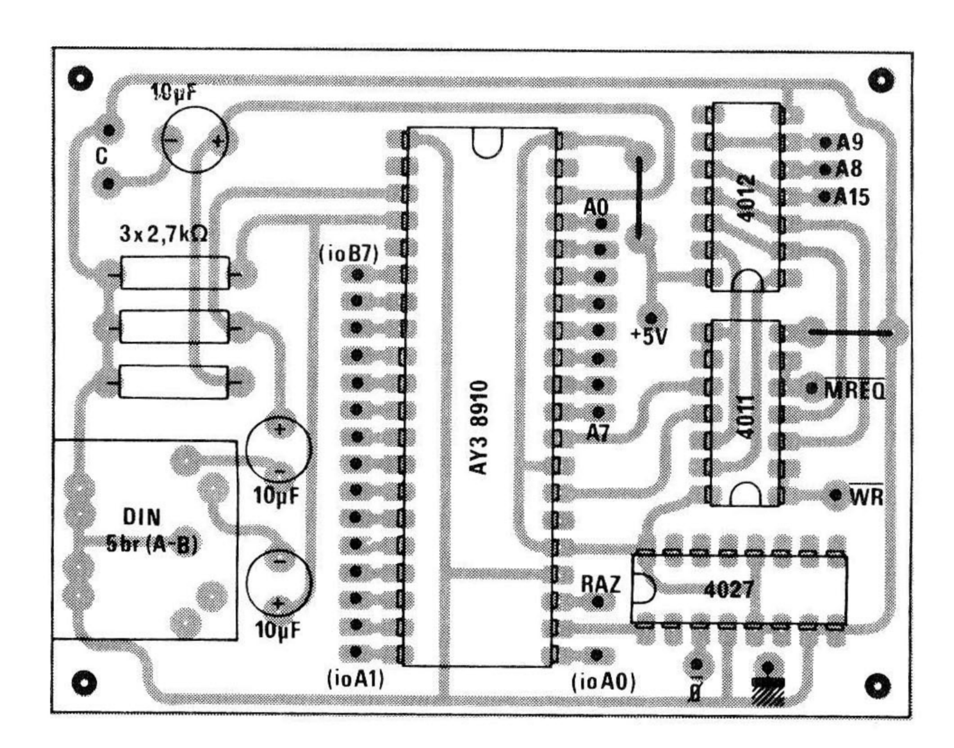


Fig. 7-5.

- permettre le transfert sur tout autre système à Z 80 par simple remplacement du connecteur ;
- rendre le montage compatible avec les divers boîtiers susceptibles d'abriter le ZX 81 (montage du synthétiseur à l'intérieur du boîtier sans connecteur).

Un ou deux torons de fil ou une longueur de câble en nappe (pas plus de vingt centimètres) assureront la liaison entre la carte principale et le

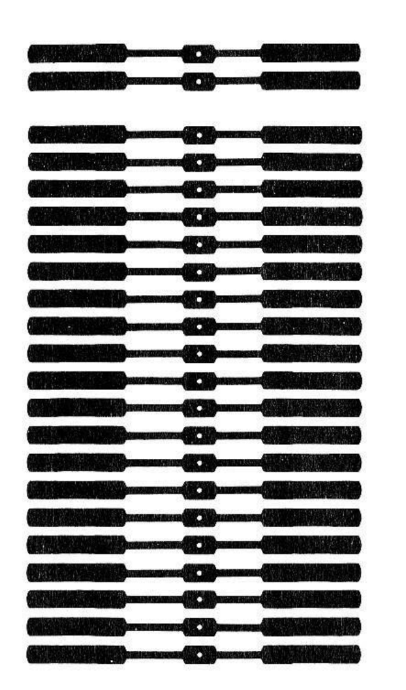
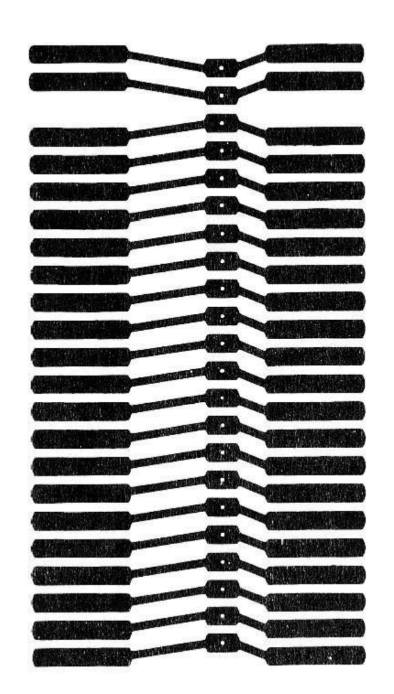



Fig. 7-6.

connecteur. Une solution intéressante consiste à utiliser le *wrapping* pour raccorder le connecteur : plusieurs extensions pourraient alors se partager le même branchement.

L'embase femelle à deux fois vingt-deux broches (plus le détrompeur) devra être réalisée par sciage soigneux d'un connecteur légèrement plus grand, car il ne s'agit naturellement pas d'un modèle standard ! Le détrompeur sera constitué d'un petit morceau de plastique ou d'époxy *non cuivré* collé à la place de deux contacts extraits avec soin.

Au niveau du raccordement BF, il a été prévu une prise DIN recevant les voies A et B sur les deux canaux stéréo. Une sortie séparée a été réservée à la voie C, qui pourrait par exemple attaquer un petit transducteur piézo-électrique ou un ampli incorporé de petite puissance, alimenté par le + 9 V. de l'ordinateur.

Pour un usage monophonique, il est facile d'ajouter deux points de soudure assurant la mise en parallèle des trois voies au niveau des résistances de 2,7 k.

Le niveau disponible est compatible avec toute entrée « ligne 0 dB » d'amplificateur.

Lors du câblage, on notera que les seize points repérés I0A0 à I0B7 ne sont pas utilisés en fonctionnement normal (ports d'entrée-sortie que nos lecteurs pourront exploiter par ailleurs).

Remarque importante : sélection des composants

Notre montage a été étudié autour d'un circuit spécialisé de Général Instrument, le AY-3-8910 bien connu pour équiper une grande majorité d'ordinateurs individuels.

Ce composant offre une incomparable palette d'effets sonores, mais fonctionne un peu trop lentement pour le ZX 81.

La solution adoptée par les fabricants des cartes sonores du commerce pour réaliser l'adaptation consiste à intercaler une batterie de bascules « latch » en supplément de celles déjà présentes dans le synthétiseur lui-même.

Ainsi, QUICKSILVA arrive à un total de six boîtiers, dont certains peuvent poser des problèmes de disponibilité à l'amateur.

Pour notre part, nous avons décidé de « jouer la simplicité » en mettant en œuvre différents artifices décrits dans les pages précédentes. Résultat, trois boîtiers seulement sont utilisés en plus du synthétiseur, et il s'agit de circuits CMOS des plus courants.

Cependant, le matériel est utilisé à l'extrême limite de ses possibilités, et toute liberté prise avec la qualité des circuits intégrés employés entraîne un mutisme farouche du montage !

Sélection des circuits CMOS

Que nos lecteurs se rassurent, tout au long de nos essais, nous n'avons jamais eu à incriminer un AY-3-8910. Il s'agit d'un composant de haute qualité, dont les caractéristiques se maintiennent dans de très étroites tolérances. Il n'en va hélas pas de même avec les circuits CMOS, car les différents fabricants utilisent des procédés de fabrication extrêmement variables.

En ce qui concerne le 4027, la plupart des marques donnent satisfaction, à condition d'éviter certains revendeurs peu scrupuleux, se fournissant parfois à des sources inavouables. En cas de doute, le contrôle est immédiat pour qui dispose d'un oscilloscope : la fréquence des signaux de sortie de la bascule doit être exactement moitié de celle de l'entrée, et parfaitement stable.

Notre maquette utilise un HCF 4027 BE de SGS-ATES, mais les meilleurs résultats sont à attendre des marques suivantes : RTC, SIGNETICS, FAIRCHILD, dont les limites de fréquence sont près de dix fois plus élevées que celles de la concurrence (procédé LOCMOS ou équivalents).

C'est cependant au niveau du 4012 et surtout du 4011 que se pose la majorité des problèmes.

Les signaux présents sur les bus du ZX 81 sont extrêmement brefs (quelques centaines de nanosecondes), et ne peuvent être visualisés à l'oscilloscope qu'au prix d'un soin extrême dans les réglages de base de temps et de synchronisation. A la limite, un oscilloscope de qualité douteuse donnera l'illusion d'une complète absence de signaux.

Un synchronisme rigoureux est indispensable entre les impulsions attaquant le synthétiseur, sous peine d'échec des commandes émanant de l'ordinateur.

Il importe donc que le 4011 comme le 4012 soient choisis de très bonne qualité, en attendant la disponibilité de la nouvelle famille HCMOS, beaucoup plus rapide.

La vérification consiste à mesurer, pour chaque porte du boîtier, la résistance apparaissant entre une entrée et la sortie. On ne retiendra que les 4011 donnant plus de 5 mégohms, et les 4012 se situant à plus de 4 mégohms.

L'expérience montre que de tels « oiseaux rares » existent dans la plupart des « fonds de tiroirs » d'amateurs.

A défaut, on pourra, vu le faible prix et l'usage fréquent de ces composants, acheter un petit échantillonnage dans diverses marques, avec une préférence pour NS, RCA, et Fairchild. De toute façon, le coût de cette sélection restera inférieur à celui des composants nécessaires à la mise en œuvre d'un principe moins délicat. En dernier recours, on peut aussi songer à passer en TTL-LS, au prix de quelques modifications de brochage.

La programmation du système

Une fois relié au ZX 81, le générateur de sons doit rester strictement muet, même en cas d'exécution de programmes classiques. Seuls quelques rares ordres POKE pourraient avoir une action sur le synthétiseur, mais de tels ordres devraient agir dans des zones de mémoire inutilisées par les programmes BASIC 1 K ou 16 K.

L'émission intempestive de sons ou, pire encore, le blocage du ZX 81 (pas de curseur à la mise sous tension) seraient les indices d'une faute de câblage dans le générateur ou son connecteur, de nature à mettre éventuellement en danger certains composants de l'ordinateur.

Pour « donner la vie » au générateur de sons, il est nécessaire d'écrire des programmes très spécifiques, tels que ceux qui vont suivre.

Le petit programme de la *figure 7-7* constitue la base de départ indispensable : ses lignes 2 à 5 viennent charger la routine machine dans la ligne 1, qu'il ne faut donc pas omettre ou même modifier les yeux

```
   1 REM ROUTINE DE BASE PSG
   2 LET A$="0062540620000500001
30050000129005200195134064"
   3 FOR F=1 TO LEN A$/3
   4 POKE 16513+F,VAL A$((3*F)-2
 TO 3*F)
   5 NEXT F
  10 REM **********************
  20 REM
1000 REM **********************
1009 POKE 16519,R
1012 POKE 16522,D
1013 LET L=USR 16514
1014 LET L=USR 16514
1015 RETURN
1016 REM COPYRIGHT1983
```

Fig. 7-7.

fermés. La plus grande attention est de rigueur lors de la frappe de la ligne 2, constituée de la suite des codes décimaux fournis par la figure 7-3.

Ce programme n'est pas utilisable seul, car il faut lui ajouter, entre les lignes 10 et 1 000, la liste exhaustive des interventions à exécuter sur les registres du générateur, en respectant la présentation suivante :

— mettre le numéro du registre dans la variable R ;

— mettre la donnée que l'on souhaite introduire dans ce registre, dans la variable D ;

— faire GOSUB 1 000 ;

— et ainsi de suite pour chaque registre sur le contenu duquel il faut intervenir.

A titre de démonstration, le programme de la *figure 7-8* utilise la fonction RND du BASIC pour balayer tous les registres intéressants avec tous les contenus possibles. Sous réserve d'un fonctionnement suffisamment prolongé, on pourra ainsi bénéficier d'un aperçu à peu près complet des possibilités du système. Ne pas se formaliser de périodes silencieuses pouvant durer un certain temps, car cela fait partie des modes de fonctionnement prévus.

Le programme de la *figure 7-9* est une variante se limitant à des émissions aléatoires de tonalités de niveau fixe : un exemple des talents artistiques du ZX 81 livré à lui-même ! Avec le programme de la *figure 7-10*, nous abordons véritablement la commande directe du générateur par l'utilisateur. Ce programme permet donc la découverte progressive et réfléchie des possibilités très étendues du système, et la

```
   1 REM DEMONSTRATION PSG
   2 LET A$="006254062000005000001
30050000129005200195134064"
   3 FOR F=1 TO LEN A$/3
   4 POKE 16513+F,VAL A$((3*F)-2
 TO 3*F)
   5 NEXT F
  10 REM ***********************
  20 LET R=RND*13
  30 LET D=RND*255
  40 GOSUB 1000
  50 GOTO 20
1000 REM ***********************
1009 POKE 16519,R
1012 POKE 16522,D
1013 LET L=USR 16514
1014 LET L=USR 16514
1015 RETURN
1016 REM COPYRIGHT1983
```

Fig. 7-8.

```
   1 REM DEMONSTRATION PSG
   2 LET A$="006254062000005000001
30050000129005200195134064"
   3 FOR F=1 TO LEN A$/3
   4 POKE 16513+F,VAL A$((3*F)-2
 TO 3*F)
   5 NEXT F
  10 REM ***********************
  20 LET R=8
  30 LET D=12
  40 GOSUB 1000
  50 LET R=7
  60 LET D=62
  70 GOSUB 1000
  80 LET R=1
  90 LET D=RND*15
 100 GOSUB 1000
 110 GOTO 20
1000 REM ***********************
1009 POKE 16519,R
1012 POKE 16522,D
1013 LET L=USR 16514
1014 LET L=USR 16514
1015 RETURN
1016 REM COPYRIGHT1983
```

Fig. 7-9.

```
   1 REM PROGRAMMATION PSG
   2 LET A$="00625406200000500001
300050000129000520001951340 64"
   3 FOR F=1 TO LEN A$/3
   4 POKE 16513+F,VAL A$((3*F)-2
 TO 3*F)
   5 NEXT F
   6 CLS
   7 PRINT AT 10,5;"NO REGISTRE
?"
   8 INPUT R
   9 POKE 16519,R
  10 PRINT AT 10,5;"CONTENU DU R
EGISTRE ";R;" ?"
  11 INPUT D
  12 POKE 16522,D
  13 RAND USR 16514
  14 RAND USR 16514
  15 GOTO 6
  16 REM COPYRIGHT1983
```

Fig. 7-10.

mise au point d'effets jugés intéressants, dont il suffira de noter les paramètres pour pouvoir les reproduire par la suite. Eventuellement, quelques lignes de plus pourraient être prévues pour consigner sur imprimante la succession des tentatives effectuées.

Arrivant à ce stade, il est nécessaire de traiter dans le détail de la structure des ordres devant être introduits dans les différents registres.

Sélection des voies

La *figure 7-11* est le « poste d'aiguillage » de tout le système : elle indique, pour chaque possibilité de chargement du *registre 7* (parmi 64 variantes), si chacune des trois voies A, B, ou C est reliée à son générateur de tonalités, au générateur de bruit commun, aux deux à la fois, ou à aucun des deux dispositifs.

Pour simplifier les choses, nous conseillons à nos lecteurs de débuter leurs expérimentations avec la seule voie A, et en lui faisant émettre soit une tonalité, soit un bruit, mais pas les deux à la fois. Les contenus respectifs du registre 7 sont alors 62 et 55.

Cette manœuvre ne suffit cependant pas à faire émettre un son à la voie sélectionnée : encore faut-il lui assigner un niveau de sortie (fixe ou variable) et une fréquence tombant dans le spectre audible (le circuit peut délivrer des fréquences atteignant 100 kHz !).

R7	VOIES BRUIT			VOIES TONALITE		
0	A	B	C	A	B	C
1	A	B	C		B	C
2	A	B	C	A		C
3	A	B	C			C
4	A	B	C	A	B	
5	A	B	C		B	
6	A	B	C	A		
7	A	B	C			
8		B	C	A	B	C
9		B	C		B	C
10		B	C	A		C
11		B	C			C
12		B	C	A	B	
13		B	C		B	
14		B	C	A		
15		B	C			
16	A		C	A	B	C
17	A		C		B	C
18	A		C	A		C
19	A		C			C
20	A		C	A	B	
21	A		C		B	
22	A		C	A		
23	A		C			
24			C	A	B	C
25			C		B	C
26			C	A		C
27			C			C
28			C	A	B	
29			C		B	
30			C	A		
31			C			
32	A	B		A	B	C
33	A	B			B	C
34	A	B		A		C
35	A	B				C
36	A	B		A	B	
37	A	B			B	
38	A	B		A		
39	A	B				
40		B		A	B	C
41		B			B	C
42		B		A		C
43		B				C
44		B		A	B	
45		B			B	
46		B		A		
47		B				
48	A			A	B	C
49	A				B	C
50	A			A		C
51	A					C
52	A			A	B	
53	A				B	
54	A			A		
55	A					
56				A	B	C
57					B	C
58				A		C
59						C
60				A	B	
61					B	
62				A		
63						

Fig. 7-11.

Réglage du niveau de sortie

Chaque voie possède un atténuateur de sortie réglable en seize niveaux notés de 0 à 15, et correspondant aux tensions indiquées par la *figure 7-13*. La variation est logarithmique, et s'adapte donc très bien aux imperfections de l'oreille.

Pour un réglage donné de l'amplificateur externe, on peut ainsi aller, par programmation, du son à peine audible au maximum supportable.

Le *tableau A* de la *figure 7-12* regroupe ces données, tout en précisant que les voies A, B, C utilisent à cet effet les registres 8, 9, 10.

Charger la valeur 16 dans ces registres revient à neutraliser la commande directe du niveau, et à subordonner celui-ci au *générateur d'enveloppes*.

Le générateur d'enveloppes

Quel que soit sa programmation, le générateur d'enveloppes n'agit que sur la ou les voies qui lui sont connectées par le biais de la mise à 16 de leur registre d'amplitude. Rappelons que les trois voies se partagent le même générateur d'enveloppes.

Le *tableau B* de la *figure 7-12* montre que ce générateur peut moduler l'amplitude, soit de façon permanente (variations en dents de scie de diverses sortes), soit de façon ponctuelle (effet de percussion), avec deux effets spéciaux notés 11 et 13.

En mode percussion, il faut recharger le *registre 13* chaque fois que l'effet doit être renouvelé, alors qu'en mode cyclique, l'effet durera tant qu'un contre-ordre n'aura pas été donné.

Il est important de pouvoir ajuster soit la durée de la percussion, soit la période du cycle. Ce réglage peut être effectué sur 65 536 niveaux, au moyen de deux registres acceptant des contenus compris entre 0 et 255. Le *registre 12* sert au réglage grossier, et le *registre 11* au réglage fin (256 pas du registre 11 comblent l'écart existant entre deux pas du registre 12).

Contrôle de volume :

Ø = extinction totale à 15 = volume maxi
16 = commande par le générateur d'enveloppes

Tableau A

voie	A	B	C
registre	8	9	1Ø

Fig. 7-12.
suite au verso

Contrôle de fréquence :

- *bruit :* registre 6 commun aux 3 voies 1 (sec) à 31 (sourd)
- *tonalités :*

Tableau B

voie	A	B	C
registre grossier (Ø à 15)	1	3	5
registre fin (Ø à 255)	Ø	2	4

Ø = aigu 15 = grave   Ø = aigu 255 = grave

Sélection d'enveloppes : (générateur commun aux 3 voies)

- *réglage fréquence ou durée :*

Tableau C

grossier	registre 12 (Ø à 255)
fin	registre 11 (Ø à 255)

Ø = court
255 = long

- *réglage de forme :* registre 13 voir figures ci-dessous

Tableau D

Ø ou 9
4 ou 15
11
13

recharger le registre 13 pour renouveler l'effet

8
1Ø ou 14
12

effet permanent

Fig. 7-12.

Réglage des fréquences

En matière de tonalités, la fréquence conditionne bien sûr la hauteur du son ou de la note que l'on émet. Avec la fréquence d'horloge utilisée, la gamme couverte va de 25 Hz à 100 kHz, soit environ huit octaves pour les applications musicales !

En ce qui concerne le bruit, on peut moins facilement parler de fréquence. On conviendra qu'un bruit de haute fréquence est SEC tandis qu'un bruit de basse fréquence est SOURD. La sonorité du bruit est programmée dans le *registre 6*, le bruit le plus sec correspondant à la valeur 1, et le plus sourd à la valeur 31.

Pour les tonalités, un système à deux registres est mis en œuvre, que résume le *tableau C* de la *figure 7-12*.

Le réglage grossier n'opère qu'entre 0 et 15, alors que le réglage fin couvre la plage de 0 à 255. La voie A utilise les registres 0 et 1, la voie B les registres 2 et 3, et la voie C les registres 4 et 5. Le son émis sera d'autant plus aigu que le contenu des registres prendra une valeur faible. Signalons toutefois que la valeur zéro conduit à la neutralisation du bruit, ou de la tonalité si les deux registres de fréquence sont affectés.

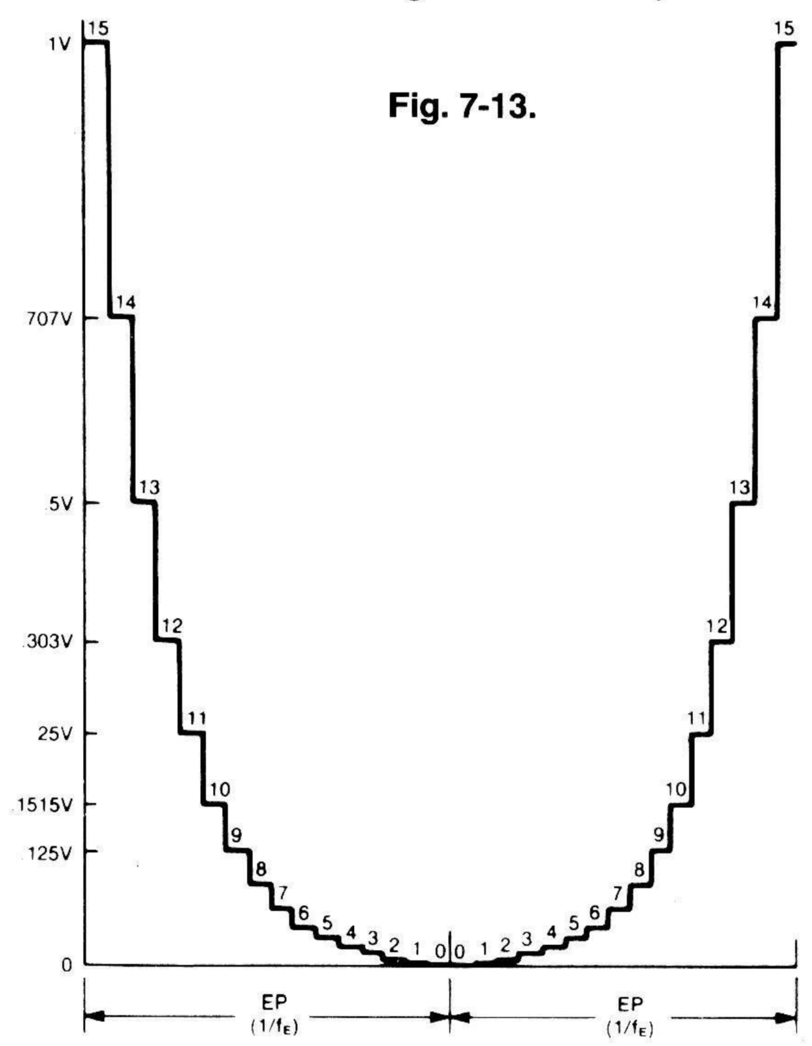


Fig. 7-13.

Fig. 7-14.

NOTE	OCTAVE	FREQUENCE	VOIE A: / B: / C:	REGISTRE R1 / R3 / R5	R0 / R2 / R4
DO	1	32.7		12	33
DO=	1	34.64		11	115
RE	1	36.7		10	206
RE=	1	38.89		10	51
MI	1	41.2		9	160
FA	1	43.65		9	22
FA=	1	46.24		8	147
SOL	1	48.99		8	24
SOL=	1	51.91		7	164
LA	1	54.99		7	54
LA=	1	58.27		6	206
SI	1	61.73		6	109
DO	2	65.4		6	16
DO=	2	69.29		5	185
RE	2	73.41		5	103
RE=	2	77.78		5	25
MI	2	82.4		4	208
FA	2	87.3		4	139
FA=	2	92.49		4	73
SOL	2	97.99		4	12
SOL=	2	103.82		3	210
LA	2	109.99		3	155
LA=	2	116.54		3	103
SI	2	123.47		3	54
DO	3	130.81		3	8
DO=	3	138.59		2	220
RE	3	146.83		2	179
RE=	3	155.56		2	140
MI	3	164.81		2	104
FA	3	174.61		2	69
FA=	3	184.99		2	36
SOL	3	195.99		2	6
SOL=	3	207.65		1	233
LA	3	219.99		1	205
LA=	3	233.08		1	179
SI	3	246.94		1	155
DO	4	261.62		1	132
DO=	4	277.18		1	110
RE	4	293.66		1	89
RE=	4	311.12		1	70
MI	4	329.62		1	52
FA	4	349.22		1	34
FA=	4	369.99		1	18
SOL	4	391.99		1	3
SOL=	4	415.3		0	244
LA	4	439.99		0	230
LA=	4	466.16		0	217
SI	4	493.88		0	205

Fig. 7-14.

NOTE	OCTAVE	FREQUENCE	VOIE A: B: C:	REGISTRE R1 R3 R5	REGISTRE R0 R2 R4
DO	5	523.24		0	194
DO=	5	554.36		0	183
RE	5	587.32		0	172
RE=	5	622.25		0	163
MI	5	659.25		0	154
FA	5	698.45		0	145
FA=	5	739.98		0	137
SOL	5	783.98		0	129
SOL=	5	830.6		0	122
LA	5	879.99		0	115
LA=	5	932.32		0	108
SI	5	987.76		0	102
DO	6	1046.49		0	97
DO=	6	1108.72		0	91
RE	6	1174.65		0	86
RE=	6	1244.5		0	81
MI	6	1318.5		0	77
FA	6	1396.9		0	72
FA=	6	1479.96		0	68
SOL	6	1567.97		0	64
SOL=	6	1661.2		0	61
LA	6	1759.98		0	57
LA=	6	1864.64		0	54
SI	6	1975.52		0	51
DO	7	2092.99		0	48
DO=	7	2217.44		0	45
RE	7	2349.3		0	43
RE=	7	2489		0	40
MI	7	2637		0	38
FA	7	2793.8		0	36
FA=	7	2959.93		0	34
SOL	7	3135.94		0	32
SOL=	7	3322.41		0	30
LA	7	3519.97		0	28
LA=	7	3729.28		0	27
SI	7	3951.04		0	25
DO	8	4185.98		0	24
DO=	8	4434.89		0	22
RE	8	4698.6		0	21
RE=	8	4978		0	20
MI	8	5274		0	19
FA	8	5587.61		0	18
FA=	8	5919.87		0	17
SOL	8	6271.88		0	16
SOL=	8	6644.83		0	15
LA	8	7039.95		0	14
LA=	8	7458.57		0	13
SI	8	7902.08		0	12

Quelques applications pratiques

L'une des principales utilisations d'un synthétiseur de sons est bien évidemment la reconstitution de mélodies musicales. Pour ce faire, le générateur de tonalités doit se conformer à un certain « plan de fréquences » constituant ce que les musiciens appellent une *gamme.*

Le tableau de la *figure 7-14* fournit, demi-ton par demi-ton, et sur huit octaves consécutives, les données à introduire dans chacun des deux registres de fréquence de la voie utilisée. Pour information, la fréquence obtenue est également indiquée. La justesse des notes obtenues est tout à fait satisfaisante, et il est facile de s'en convaincre en lançant le petit programme de la *figure 7-15.*

```
   1 REM GAMME AUTOMATIQUE
   2 LET A$="0052540520000500001
3005000129005200195134064"
   3 FOR F=1 TO LEN A$/3
   4 POKE 16513+F,VAL A$((3*F)-2
 TO 3*F)
   5 NEXT F
  10 REM ***********************
  20 LET R=8
  30 LET D=15
  40 GOSUB 1000
  50 LET R=7
  60 LET D=62
  70 GOSUB 1000
  80 LET R=1
  90 LET D=0
 100 GOSUB 1000
 110 LET B$="1941721541451291151
02097"
 120 FOR F=1 TO LEN B$/3
 130 LET R=0
 140 LET D=VAL B$((3*F)-2 TO 3*F
)
 150 GOSUB 1000
 155 PAUSE 100
 160 NEXT F
 170 RUN
1000 REM ***********************
1009 POKE 16519,R
1012 POKE 16522,D
1013 LET L=USR 16514
1014 LET L=USR 16514
1015 RETURN
1016 REM COPYRIGHT1983
```

Fig. 7-15.

Ce programme joue de façon ininterrompue la gamme naturelle de la cinquième octave, à un tempo fixé par la pause de la ligne 155. Il serait facile de greffer des modifications permettant de varier les effets obtenus :

- introduction d'une sonorité de percussion, plus proche de la réalité instrumentale, en mettant en service le générateur d'enveloppes ;
- obtention d'un son plus ample en réalisant des accords par mise en service d'une ou deux voies supplémentaires décalées en fréquence ;
- programmation d'une mélodie plus agréable que la gamme par simple modification du contenu de la chaîne B$ en accord avec les valeurs de la *figure 7-14* (cinquième octave et suivantes) ;
- introduction de silences de la durée d'une note en prévoyant des 000 dans B$ (correspondant à des tonalités sortant du spectre audible) ;
- effets spéciaux obtenus par mise en service simultanée du générateur de bruit, etc.

Les talents musicaux de ce générateur ne doivent pas faire oublier pour autant ses possibilités de bruiteur.

```
   1 REM SONS PREPROGRAMMES
   2 LET A$="00625406200005000001
200500001290052001951340 64"
   3 FOR F=1 TO LEN A$/3
   4 POKE 16513+F,VAL A$((3*F)-2
 TO 3*F)
   5 NEXT F
  10 REM ************************
  20 IF INKEY$="1" THEN GOTO 200
  30 IF INKEY$="2" THEN GOTO 300
  40 IF INKEY$="3" THEN GOTO 400
  50 IF INKEY$="4" THEN GOTO 500
  60 IF INKEY$="5" THEN GOTO 600
  70 IF INKEY$="6" THEN GOTO 700
  80 IF INKEY$="7" THEN GOTO 800
  90 IF INKEY$="8" THEN GOTO 900
 100 IF INKEY$="9" THEN GOSUB 10
00
 110 IF INKEY$="0" THEN GOSUB 10
20
 190 GOTO 10
 200 LET R=8
 205 LET D=15
 210 GOSUB 1000
 215 LET R=7
 220 LET D=62
 225 GOSUB 1000
 230 LET R=1
 235 LET D=0
 240 GOSUB 1000
 241 LET R=0
 242 LET D=100
 243 GOSUB 1000
 245 LET R=12
```

Fig. 7-16.
suite au verso

```
250 LET D=2
255 GOSUB 1000
260 LET R=13
265 LET D=0
270 GOSUB 1000
280 GOTO 980
300 LET R=8
305 LET D=16
310 GOSUB 1000
315 LET R=7
320 LET D=62
325 GOSUB 1000
330 LET R=1
335 LET D=3
340 GOSUB 1000
345 LET R=12
350 LET D=50
355 GOSUB 1000
360 LET R=13
365 LET D=0
370 GOSUB 1000
380 GOTO 980
400 LET R=8
405 LET D=16
410 GOSUB 1000
415 LET R=7
420 LET D=55
425 GOSUB 1000
430 LET R=6
435 LET D=1
440 GOSUB 1000
445 LET R=12
450 LET D=5
455 GOSUB 1000
460 LET R=13
465 LET D=0
470 GOSUB 1000
480 GOTO 980
500 LET R=8
505 LET D=16
510 GOSUB 1000
515 LET R=7
520 LET D=55
525 GOSUB 1000
530 LET R=6
535 LET D=31
540 GOSUB 1000
545 LET R=12
550 LET D=50
555 GOSUB 1000
560 LET R=13
565 LET D=0
570 GOSUB 1000
580 GOTO 980
600 LET R=8
605 LET D=16
610 GOSUB 1000
615 LET R=7
620 LET D=55
625 GOSUB 1000
630 LET R=6
635 LET D=1
640 GOSUB 1000
645 LET R=12
650 LET D=10
655 GOSUB 1000
660 LET R=13
665 LET D=12
670 GOSUB 1000
680 GOTO 980
980 PAUSE 20
990 GOTO 10
1000 REM ***************
1009 POKE 16519,R
1012 POKE 16522,D
1013 LET L=USR 16514
1014 LET L=USR 16514
1015 RETURN
1016 REM COPYRIGHT 1983
1020 FOR F=0 TO 13
1030 LET R=F
1040 LET D=0
1050 GOSUB 1000
1060 NEXT F
1070 RETURN
```

Fig. 7-16.

Le programme de la *figure 7-16* permet d'expérimenter divers sons « préprogrammés » pouvant être « appelés » par simple pression sur les touches numériques du ZX 81. Notons que les touches 6, 7 et 8 sont réservées à des sons librement définis par l'utilisateur, les sous-programmes correspondants devant être implantés respectivement à partir des lignes 700, 800 et 900.

Les touches 9 et Ø jouent des rôles particuliers :

• La touche 9 permet de répéter le dernier bruit émis, mais avec un temps de réponse considérablement écourté, puisque les variables sont déjà chargés, et qu'il suffit d'appeler la routine en langage machine. L'emploi adroit de cette touche permet d'obtenir des résultats assez saisissants.

• La touche Ø déclenche l'effacement de tous les registres du générateur et présente donc, en moins rapide, le même fonctionnement que le bouton de remise à zéro prévu dans le circuit du générateur.

Bien évidemment, tous les sous-programmes appelés ici à partir du clavier par des fonctions INKEY$ peuvent être utilisés à partir d'un programme, notamment de jeu, que l'on souhaiterait « sonoriser ». Il faudrait alors utiliser des GOSUB à la place des GOTO, et implanter des RETURN à la place des GOTO 980.

Un synthétiseur de laboratoire

Le programme de la *figure 7-17* suffirait à lui seul à rentabiliser une fois pour toutes l'achat d'un ZX 81 et la construction du générateur de sons !

Il fournit en effet un synthétiseur de fréquence capable de générer avec la précision du quartz (ou tout au moins du filtre céramique 6,5 MHz du ZX 81), toute une gamme de fréquences comprises entre 25 Hz et 100 kHz. En fait, les meilleures performances sont obtenues pour les fréquences les plus basses et, aux alentours de 1 000 Hz, ce synthétiseur peut se mesurer aux meilleurs générateurs.

L'utilisation est extrêmement simple, puisqu'il suffit de répondre aux questions de la machine par une valeur de fréquence en Hz, puis de valider par NEWLINE.

Si la valeur entrée n'est pas valide, la machine la refuse, mais dans le cas contraire, elle la rappelle à l'écran avec un arrondi évitant de prendre trop au sérieux des chiffres qui pourraient n'être pas tous significatifs. Aussitôt, la fréquence choisie est générée par la voie A, jusqu'à ce qu'une nouvelle valeur soit entrée.

Les applications d'un tel synthétiseur sont innombrables dans le laboratoire de l'amateur, et s'étendent jusqu'à la vérification de l'audition dans les aiguës des amateurs d'amplis HIFI taquinant le mégahertz ! Des surprises pas toujours très flatteuses sont garanties...

Il pourrait être avantageux de réduire l'encombrement mémoire de ce programme en supprimant les lignes génératrices du code machine une fois la ligne REM construite, en simplifiant le dialogue, et en utilisant les artifices habituels au niveau des constantes numériques.

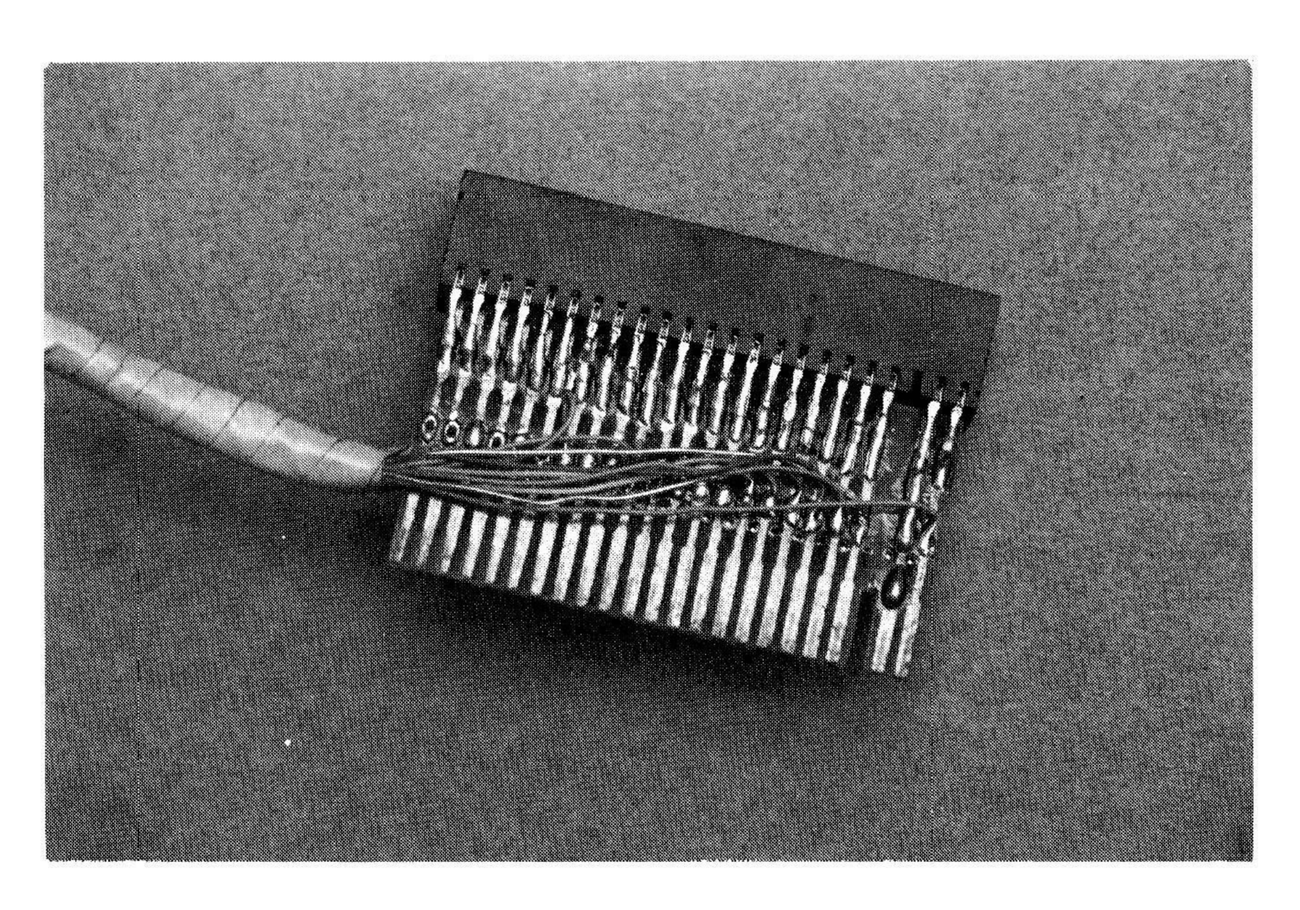
```
   1 REM SYNTHETISEUR DE FREQUEN
CE
   2 LET A$="0062540620000500001
3000500001290005200019513406 4"
   3 FOR F=1 TO LEN A$/3
   4 POKE 16513+F,VAL A$((3*F)-2
 TO 3*F)
   5 NEXT F
   6 GOTO 100
  10 REM ***********************
  20 LET R=6
  30 LET D=15
  40 GOSUB 1000
  50 LET R=7
  60 LET D=62
  70 GOSUB 1000
  80 LET R=1
  81 LET D=N1
  82 GOSUB 1000
  85 LET R=0
  86 LET D=N2
  90 GOSUB 1000
  93 IF INKEY$<>"" THEN GOTO 100
  95 GOTO 20
 100 CLS
 105 PRINT AT 12,5;"FREQUENCE EN
 HERTZ ?"
 106 PRINT
 107 PRINT "MINI 25 HZ   MAXI 10
0 KHZ"
 110 INPUT L
 115 IF L<25 OR L>100000 THEN GO
TO 100
 120 CLS
 140 DIM B(12)
 150 LET N=101562.5/L
 160 LET X=2048
 170 FOR Q=1 TO 12
 180 LET B(13-Q)=(N>=X)
 190 IF N>=X THEN LET N=N-X
 200 LET X=X/2
 210 NEXT Q
 220 LET N1=B(9)+2*B(10)+4*B(11)
+8*B(12)
 230 LET N2=B(1)+2*B(2)+4*B(3)+8
*B(4)+16*B(5)+32*B(6)+64*B(7)+12
8*B(8)
 232 PRINT AT 12,5;"VOICI DU ";I
NT L;" HERTZ"
 233 PRINT
 234 PRINT
 235 PRINT "POUR CHANGER,PRESSER
 UNE TOUCHE"
 240 GOSUB 10
1000 REM ***********************
```

```
1009 POKE 16519,R
1012 POKE 16522,D
1013 LET L=USR 16514
1014 LET L=USR 16514
1015 RETURN
1016 REM COPYRIGHT1983
```

Fig. 7-17.

Ainsi, le logiciel pourrait être figé dans une EPROM, ce qui faciliterait notablement la mise en œuvre rapide du synthétiseur. A la limite, l'afficheur miniature précédemment décrit pourrait même remplacer l'écran TV, au prix de quelques précautions en matière d'adresses mémoire utilisées.

Un procédé similaire, associé au remplacement du clavier par des touches appropriées, pourrait servir de base à la construction d'un instrument de musique d'emploi extrêmement souple, puisque intégralement programmable...



Ce connecteur « gigogne » à réaliser par soi-même est la pièce maîtresse de la plupart des extensions pour ZX 81.

Chapitre 8

Des applications pratiques



Arrivés à ce stade de la lecture de cet ouvrage, nos lecteurs disposent d'une panoplie complète d'outils leur permettant de « figer » un ZX 81 dans une application pratique, avec un maximum de confort d'utilisation.

Un tel choix n'est cependant nullement définitif : la plupart des applications envisageables nécessitent pratiquement les mêmes circuits

d'entrée-sortie, seul le logiciel implanté en EPROM venant déterminer le comportement effectif du système.

Tel est l'avantage essentiel de l'utilisation de microprocesseurs : une extrême souplesse de mise en œuvre.

Le simple remplacement d'une EPROM sur un support permet de transformer, toutes proportions gardées, un ascenseur en machine à laver ou en instrument de musique !

Cela, bien évidemment, au niveau des circuits de commande et non des parties mécaniques...

La conséquence de cette versatilité est que chacun pourra, grâce à quelques notions de programmation en BASIC, ou mieux, en langage machine, développer facilement le système répondant à son besoin particulier.

Nous ne pouvons évidemment aller au devant de ces besoins, mais plutôt illustrer le propos qui fut le nôtre tout au long de ce livre, par des exemples concrets de réalisations que nous avons menées à bien.

Ces exemples seront choisis de telle façon que les routines utilisées puissent sans grandes difficultés être incorporées dans les futures réalisations de nos lecteurs.

Un système de commande de feux de circulation

Le court logiciel machine (81 octets) proposé ici permet d'utiliser six des huit sorties d'une carte 8ES reliée à un ZX 81 (équipé ou non de la ROM Sinclair, mais muni de sa RAM 1 K), pour commander les deux séries de trois feux colorés équipant un carrefour à deux artères principales.

Une zone remplie de sept NOP (code Ø) est prévue pour l'éventuelle introduction d'une routine de lecture des entrées, qui pourrait servir à prendre en compte l'actionnement de boîtes à boutons pour piétons, de détecteurs de véhicules, ou d'un boîtier de commande manuelle (agent de police). Il faudrait alors appeler des sous-programmes supplémentaires pour lesquels il ne manque certes pas d'espace dans le reste de l'EPROM !

Avec la même carte d'interface, mais en programmant différemment la 2716, on pourrait commander un ascenseur, un réseau de trains miniatures, ou d'autres équipements...

La *figure 8-1* donne la liste complète des octets du programme « feux de circulation », avec leurs adresses, alors que la *figure 8-2* livre une

0	62	41	1
1	12	42	205
2	211	43	57
3	127	44	32
4	6	45	62
5	5	46	36
6	205	47	211
7	57	48	127
8	32	49	6
9	62	50	1
10	20	51	205
11	211	52	57
12	127	53	32
13	6	54	195
14	1	55	0
15	205	56	32
16	57	57	14
17	32	58	255
18	62	59	22
19	36	60	255
20	211	61	21
21	127	62	0
22	6	63	0
23	1	64	0
24	205	65	0
25	57	66	0
26	32	67	0
27	62	68	0
28	33	69	194
29	211	70	61
30	127	71	32
31	6	72	13
32	5	73	194
33	205	74	59
34	57	75	32
35	32	76	5
36	62	77	194
37	34	78	57
38	211	79	32
39	127	80	201
40	6		

Fig. 8-1. — *Programmation de la 2716 (en décimal) pour les « feux de circulation ».*

version désassemblée destinée à faciliter la compréhension de ce logiciel.

Signalons que les instructions d'entrée/sortie ont été prévues pour une carte utilisant le port 127 (7 Fh), et que des modifications fort simples pourraient permettre un fonctionnement avec toute carte utilisant un autre accès (par exemple les versions les plus récentes de la 8ES, qui permettent à l'utilisateur de choisir son port).

```
2000 LD   A,0C;  £
  02 OUT  7F;
  04 LD   B,05;  ▮
  06 CALL 2039
  09 LD   A,14;  =
  0B OUT  7F;
  0D LD   B,01;  ▪
  0F CALL 2039
  12 LD   A,24;  8
  14 OUT  7F;
  16 LD   B,01;  ▪
  18 CALL 2039
  1B LD   A,21;  5
  1D OUT  7F;
  1F LD   B,05;  ▮
  21 CALL 2039
  24 LD   A,22;  6
  26 OUT  7F;
  28 LD   B,01;  ▪
  2A CALL 2039
  2D LD   A,24;  8
  2F OUT  7F;
  31 LD   B,01;  ▪
  33 CALL 2039
  36 JP   2000
  39 LD   C,FF;
  3B LD   D,FF;
  3D DEC  D
  3E NOP
  3F NOP
  40 NOP
  41 NOP
  42 NOP
  43 NOP
  44 NOP
  45 JP   NZ,203D
  48 DEC  C
  49 JP   NZ,203B
  4C DEC  B
  4D JP   NZ,2039
  50 RET
```

Fig. 8-2.
Le programme « feux de circulation » désassemblé.

Un transmetteur téléphonique d'alarme

Cette seconde application est nettement plus évoluée que la précédente, puisqu'il s'agit pour le ZX 81 équipé d'une carte 8ES et d'une EPROM programmée selon nos indications, de gérer entièrement la transmission par téléphone, du déclenchement d'un système d'alarme.

On supposera que l'installation est équipée d'une centrale antivol possédant une sortie 12 volts destinée, par exemple, à une sirène, et

qu'une ligne téléphonique est disponible sur place, munie ou non d'un poste classique.

Le fonctionnement de l'automate sera alors le suivant :

- appel d'un premier numéro préprogrammé en EPROM ;
- en cas de non-réponse ou d'occupation, appel immédiat d'un second numéro ;
- en cas de nouvel insuccès, retour au premier numéro, et ainsi de suite autant de fois que nécessaire ;
- au décrochage, transmission d'un signal sonore codé facilement reconnaissable ;
- à titre de confirmation et de protection contre les faux numéros, ou les réponses par des personnes non au courant, continuation de ce cycle tant que le système n'aura pas été désarmé grâce à un appel téléphonique dirigé vers le transmetteur (un coup de sonnerie).

Ce fonctionnement est rendu possible par le circuit d'interface dont la *figure 8-3* donne un schéma général. De légères modifications pourront en effet se révéler nécessaires selon les caractéristiques exactes de la centrale d'alarme et de l'installation téléphonique existantes.

Les composants spéciaux nécessaires (notamment le transformateur de ligne et le circuit résistif de réglage du courant de ligne), pourront facilement être récupérés sur une épave de poste téléphonique S63. Avant tout raccordement, il est bien sûr nécessaire de recueillir les autorisations réglementaires. L'interface se décompose en deux parties principales : le circuit de prise de ligne et de numérotation, qui se charge en même temps de l'envoi de la tonalité de signalisation et un détecteur de sonnerie, pas toujours indispensable, puisque certaines centrales de surveillance se désarment d'elles-mêmes au bout d'un certain temps d'action de la sirène.

C'est cependant le logiciel de la *figure 8-4* qui constitue le cœur du système.

Les trois colonnes indiquent, de gauche à droite, l'adresse de chaque octet dans l'EPROM, l'équivalent décimal de cet octet et, pour information seulement, les adresses ayant servi à l'assemblage sur le ZX 81.

Les deux numéros de téléphone à six chiffres ont été « masqués » par de petits carrés noirs, puisqu'il s'agissait des numéros personnels de l'auteur, et chacun programmera à la place les numéros qu'il désirera. Attention, le chiffre zéro doit être programmé au moyen de l'octet 10, puisque ce chiffre correspond à dix coupures de ligne.

Moyennant des aménagements très simples, il serait possible de programmer davantage de numéros, à six ou sept chiffres, voire même de prévoir des pauses d'attente de la tonalité du 16 en cas de

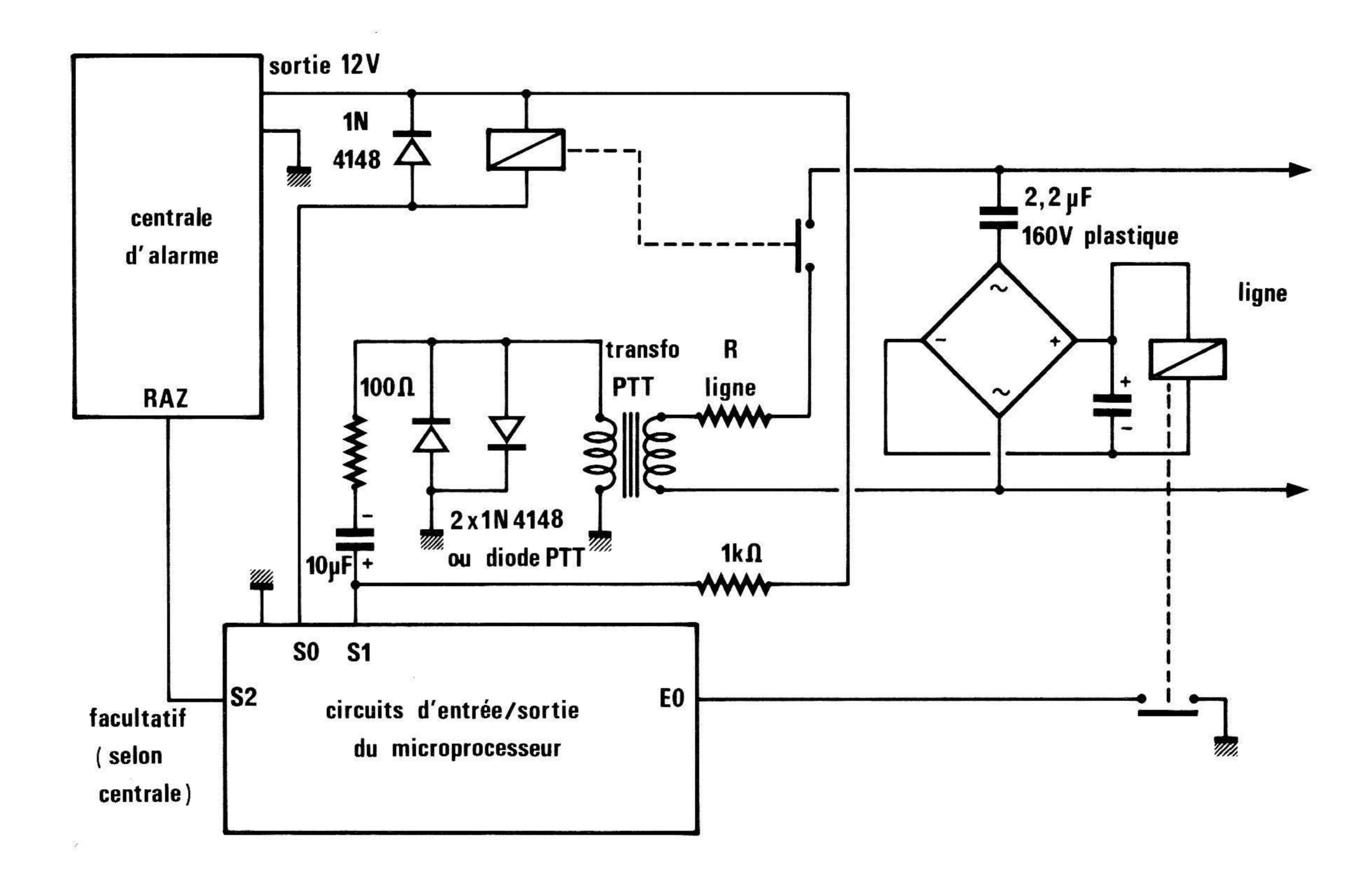


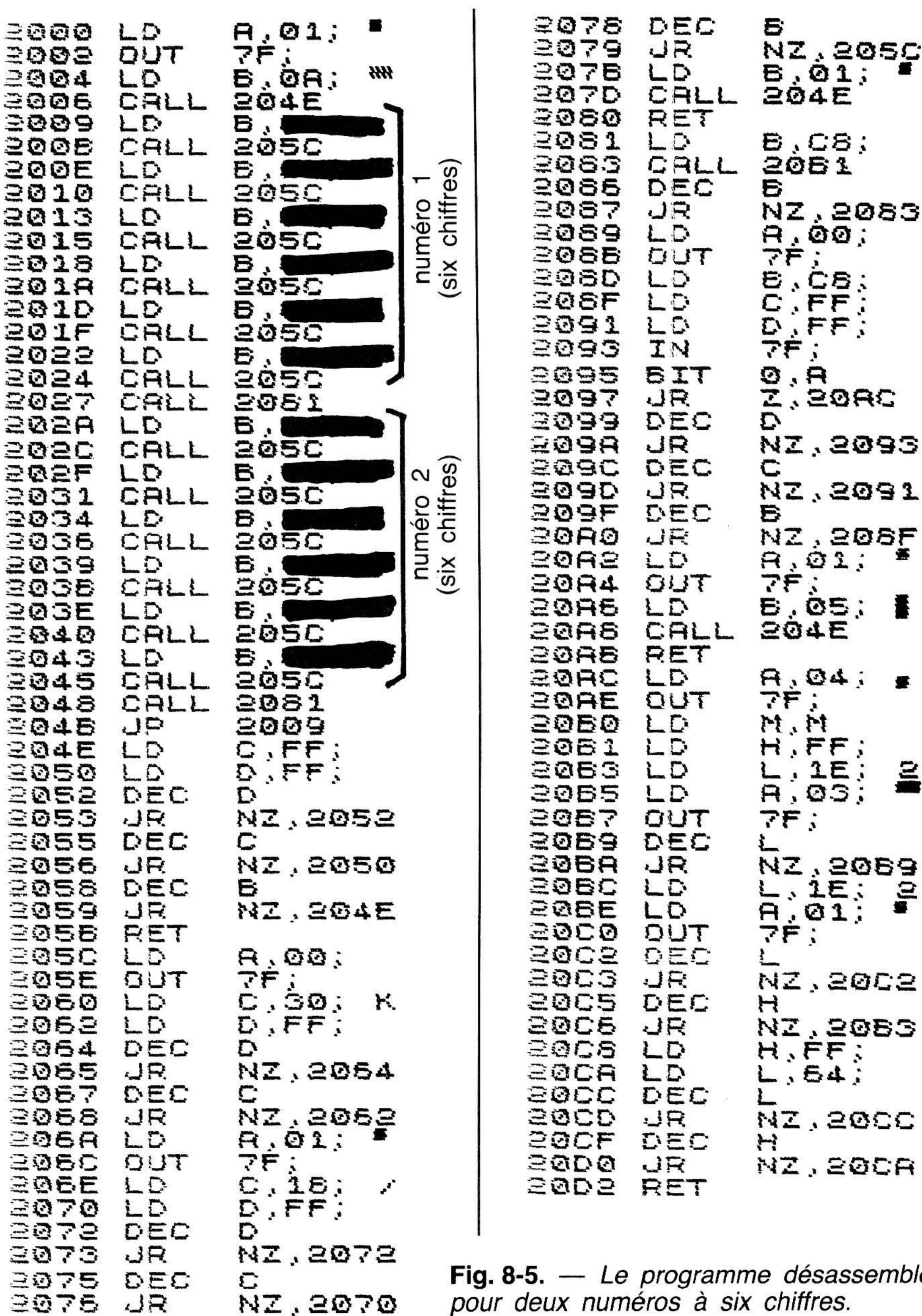
Fig. 8-3. — *Interface téléphonique.*

0	62	8192
1	1	8193
2	211	8194
3	127	8195
4	6	8196
5	10	8197
6	205	8198
7	78	8199
8	32	8200
9	6	8201
10	■	8202
11	205	8203
12	92	8204
13	32	8205
14	6	8206
15	■	8207
16	205	8208
17	92	8209
18	32	8210
19	6	8211
20	■	8212
21	205	8213
22	92	8214
23	32	8215
24	6	8216
25	■	8217
26	205	8218
27	92	8219
28	32	8220
29	6	8221
30	■	8222
31	205	8223
32	92	8224
33	32	8225
34	6	8226
35	■	8227
36	205	8228
37	92	8229
38	32	8230
39	205	8231
40	129	8232
41	32	8233
42	6	8234
43	■	8235
44	205	8236
45	92	8237
46	32	8238
47	6	8239
48	■	8240
49	205	8241
50	92	8242
51	32	8243
52	6	8244
53	■	8245
54	205	8246
55	92	8247
56	32	8248
57	6	8249
58	■	8250
59	205	8251
60	92	8252
61	32	8253
62	6	8254
63	■	8255
64	205	8256
65	92	8257
66	32	8258
67	6	8259
68	■	8260
69	205	8261
70	92	8262
71	32	8263
72	205	8264
73	129	8265
74	32	8266
75	195	8267
76	0	8268
77	32	8269
78	14	8270
79	255	8271
80	22	8272
81	255	8273
82	21	8274
83	32	8275
84	253	8276
85	13	8277
86	32	8278
87	248	8279
88	5	8280
89	32	8281
90	243	8282
91	201	8283
92	62	8284
93	0	8285
94	211	8286
95	127	8287
96	14	8288
97	48	8289
98	22	8290
99	255	8291
100	21	8292
101	32	8293
102	253	8294
103	13	8295
104	32	8296
105	248	8297
106	62	8298
107	1	8299
108	211	8300

Fig. 8-4. — *Suite au verso*

109	127	8301
110	14	8302
111	24	8303
112	22	8304
113	255	8305
114	21	8306
115	32	8307
116	253	8308
117	13	8309
118	32	8310
119	248	8311
120	5	8312
121	32	8313
122	225	8314
123	6	8315
124	1	8316
125	205	8317
126	78	8318
127	32	8319
128	201	8320
129	6	8321
130	200	8322
131	205	8323
132	177	8324
133	32	8325
134	5	8326
135	32	8327
136	250	8328
137	62	8329
138	0	8330
139	211	8331
140	127	8332
141	6	8333
142	200	8334
143	14	8335
144	255	8336
145	22	8337
146	255	8338
147	219	8339
148	127	8340
149	203	8341
150	71	8342
151	40	8343
152	19	8344
153	21	8345
154	32	8346
155	247	8347
156	13	8348
157	32	8349
158	242	8350
159	5	8351
160	32	8352
161	237	8353
162	62	8354
163	1	8355
164	211	8356
165	127	8357
166	6	8358
167	5	8359
168	205	8360
169	78	8361
170	32	8362
171	201	8363
172	62	8364
173	4	8365
174	211	8366
175	127	8367
176	118	8368
177	38	8369
178	255	8370
179	46	8371
180	30	8372
181	62	8373
182	3	8374
183	211	8375
184	127	8376
185	45	8377
186	32	8378
187	253	8379
188	46	8380
189	30	8381
190	62	8382
191	1	8383
192	211	8384
193	127	8385
194	45	8386
195	32	8387
196	253	8388
197	37	8389
198	32	8390
199	235	8391
200	38	8392
201	255	8393
202	46	8394
203	100	8395
204	45	8396
205	32	8397
206	253	8398
207	37	8399
208	32	8400
209	248	8401
210	201	8402

Fig. 8-4. — *Le contenu de l'EPROM, exprimé en code décimal. Les adresses de la colonne de gauche sont les adresses réelles, celles de la colonne de droite sont celles d'assemblage sur le ZX 81.*



```
2000 LD   A,01; ■
2002 OUT  7F;
2004 LD   B,0A; ■
2006 CALL 204E
2009 LD   B,[redacted]
200B CALL 205C
200E LD   B,[redacted]
2010 CALL 205C
2013 LD   B,[redacted]
2015 CALL 205C
2018 LD   B,[redacted]
201A CALL 205C
201D LD   B,[redacted]
201F CALL 205C
2022 LD   B,[redacted]
2024 CALL 205C
2027 CALL 2081
202A LD   B,[redacted]
202C CALL 205C
202F LD   B,[redacted]
2031 CALL 205C
2034 LD   B,[redacted]
2036 CALL 205C
2039 LD   B,[redacted]
203B CALL 205C
203E LD   B,[redacted]
2040 CALL 205C
2043 LD   B,[redacted]
2045 CALL 205C
2048 CALL 2081
204B JP   2009
204E LD   C,FF;
2050 LD   D,FF;
2052 DEC  D
2053 JR   NZ,2052
2055 DEC  C
2056 JR   NZ,2050
2058 DEC  B
2059 JR   NZ,204E
205B RET
205C LD   A,00;
205E OUT  7F;
2060 LD   C,30;  K
2062 LD   D,FF;
2064 DEC  D
2065 JR   NZ,2064
2067 DEC  C
2068 JR   NZ,2062
206A LD   A,01; ■
206C OUT  7F;
206E LD   C,18;  /
2070 LD   D,FF;
2072 DEC  D
2073 JR   NZ,2072
2075 DEC  C
2076 JR   NZ,2070
2078 DEC  B
2079 JR   NZ,205C
207B LD   B,01; ■
207D CALL 204E
2080 RET
2081 LD   B,C8;
2083 CALL 20B1
2086 DEC  B
2087 JR   NZ,2083
2089 LD   A,00;
208B OUT  7F;
208D LD   B,C8;
208F LD   C,FF;
2091 LD   D,FF;
2093 IN   7F;
2095 BIT  0,A
2097 JR   Z,20AC
2099 DEC  D
209A JR   NZ,2093
209C DEC  C
209D JR   NZ,2091
209F DEC  B
20A0 JR   NZ,208F
20A2 LD   A,01; ■
20A4 OUT  7F;
20A6 LD   B,05; ■
20A8 CALL 204E
20AB RET
20AC LD   A,04; ■
20AE OUT  7F;
20B0 LD   H,H
20B1 LD   H,FF;
20B3 LD   L,1E;  2
20B5 LD   A,03; ■
20B7 OUT  7F;
20B9 DEC  L
20BA JR   NZ,20B9
20BC LD   L,1E;  2
20BE LD   A,01; ■
20C0 OUT  7F;
20C2 DEC  L
20C3 JR   NZ,20C2
20C5 DEC  H
20C6 JR   NZ,20B3
20C8 LD   H,FF;
20CA LD   L,64;
20CC DEC  L
20CD JR   NZ,20CC
20CF DEC  H
20D0 JR   NZ,20CA
20D2 RET
```


Fig. 8-5. — *Le programme désassemblé pour deux numéros à six chiffres.*

transmission à longue distance. Il suffirait pour cela d'appeler la routine de temporisation dont nous allons découvrir l'existence.

La *figure 8-5* fournit en effet une version entièrement désassemblée, en mnémoniques Z 80 et code hexadécimal, du logiciel du transmetteur. Par ailleurs, la *figure 8-6* précise le détail des différentes routines le composant.

Nom	**adresse décimale réelle**	**adresse décimale ZX 81**	**adresse hexa ZX 81**
1er numéro	Ø	8192	2ØØØ
2e numéro	42	8234	2Ø2A
temporisation	78	8270	2Ø4E
numérotation	92	8284	2Ø5C
attente	129	8321	2Ø81
tonalité	177	8369	2ØB1

Fig. 8-6. — *Principales routines.*

Ces différentes routines s'appellent mutuellement au moyen d'instructions CALL, ce qui simplifie autant le programme que le recours à la fonction GOSUB du BASIC.

La première routine à être appelée est celle de *temporisation* : pour l'utiliser, on charge dans le registre B une valeur comprise entre 1 et 255, qui représente la durée choisie (la correspondance exacte dépend de la fréquence précise de l'horloge du système mais, d'une façon générale, on peut compter sur une seconde par unité chargée dans le registre B). Cela fait, il suffit de faire un CALL 8270 pour exécuter la temporisation (bouclage du programme pendant la durée choisie).

Cette routine sert d'abord à maintenir collé le relais de ligne pendant un temps suffisant pour que l'obtention de la tonalité puisse être considérée comme très probable. A ce moment, on charge dans B le premier chiffre du numéro à composer, puis on appelle la routine de *numérotation* par un CALL 8284. Cette routine fait battre le relais de ligne un nombre de fois égal au contenu de B, suivant le rapport cyclique normalisé 33/66 ms. Après une pause de séparation des chiffres, le programme passe au chiffre suivant.

En fin de numérotation, un CALL 8321 lance la routine d'*attente* dont le rôle est double : d'une part, laisser à l'appel le temps d'aboutir avec un nombre raisonnable de coups de sonnerie, et d'autre part l'envoi en ligne d'une tonalité hachée caractéristique, ne risquant pas d'être confondue

avec autre chose. L'émission régulière de la fréquence de 2 kHz est confiée à une routine nommée « *tonalité* », appelée depuis la routine d'attente par des CALL 8369.

Le second volet de la routine d'attente se situe après la libération de la ligne : un certain laps de temps est prévu pour permettre au destinataire de l'appel d'appeler à son tour le transmetteur. Ce faisant, la sonnerie qui en résulte est détectée au niveau de l'entrée Ø, ce qui rend immédiatement active la sortie prévue pour la remise à zéro de la centrale d'alarme. Cela fait, le microprocesseur s'arrête par exécution d'un HALT.

A défaut de cet appel en retour, le programme passe au numéro suivant, et tout le cycle recommence.

Ceux de nos lecteurs qui ne souhaiteront programmer que deux numéros à six chiffres (cas le plus fréquent, Paris n'étant pas la France !), n'auront pas à retoucher ce programme, mais seulement à y insérer les douze octets aux endroits prévus.

La programmation d'un seul numéro à sept chiffres ne posera pas non plus de problèmes : il subsistera seulement un petit espace inutilisé à la suite. Celui-ci pourrait au besoin servir à loger le préfixe 16, la pause d'attente de la tonalité interurbaine, et l'indicatif départemental d'un numéro longue distance.

Cependant, la capacité de la 2716 pourrait permettre la programmation d'un nombre considérable de numéros de toutes les longueurs. Il faudrait alors décaler d'autant vers la fin toutes les routines logées après les numéros. Cela implique seulement un renumérotage des CALL d'après la nouvelle position de chaque routine. En effet, les autres sauts, relatifs, sont tous relogeables, et le seul saut absolu renvoie au début du programme (JP 8192). Il n'a donc pas à être modifié.

La souplesse de la solution micro-informatique apparaît ici de façon éclatante, puisqu'un même circuit peut servir à composer des numéros de téléphone en nombre à peu près quelconque, selon des modalités très diverses, par de simples modifications de logiciel.

Un changement instantané de programmation (par exemple en période de vacances ou de week-end) pourrait s'effectuer par simple enfichage d'une autre EPROM !

Nous espérons que cet exemple simple quoique performant aura su convaincre nos lecteurs des avantages que présente dans ce genre de cas la solution « microprocesseur ».

N'en déduisons toutefois pas qu'il s'agit là de la panacée ! Bien des domaines restent encore exclus du champ d'action de la micro-informatique, pour des raisons de rapidité, de rentabilité, d'encombrement, et bien d'autres encore.

Si les temps sont encore loin, où l'on pourra remplacer n'importe quel circuit intégré spécifique par un microprocesseur programmé comme il convient, il faut être conscient que le domaine d'application de ces techniques s'élargit de jour en jour. Profitons-en donc dès maintenant !

Signalons pour terminer à nos lecteurs qui pourraient éprouver quelque crainte à intervenir sur des circuits téléphoniques, dont le fonctionnement est assez particulier, qu'ils pourront s'initier entièrement à ces techniques en lisant notre ouvrage *INTERPHONE TELEPHONE Montages périphériques*, paru chez le même éditeur.



Chapitre 9

Une carte microprocesseur compatible ZX 81

Remarques préliminaires

Il ne fait aucun doute que l'adjonction au ZX 81 (ou à tout ordinateur similaire), d'une carte d'entrée-sortie, ouvre la porte à une extrême variété d'applications dans le domaine des automatismes (chauffage, chemins de fer miniatures, systèmes d'alarme, etc.).

Cependant, une fois passée la phase de mise au point, il faut bien se rendre compte que le clavier, l'écran TV, l'interface cassette et autres perfectionnements ne servent plus à rien, peuvent à la limite devenir gênants, et que l'assemblage d'un ordinateur « de table » avec une carte d'interface se prête mal à l'insertion dans un ensemble « de terrain ».

Nous venons de décrire toute une gamme d'accessoires permettant de transformer un ZX 81 en un système « figé », programmé en vue d'une application spécifique.

Nous pensons cependant que les plus passionnés de nos lecteurs ne se contenteront pas d'une application unique, et qu'un seul ZX 81 ainsi adapté ne leur suffira pas !

Il y a de nombreux avantages à mettre au point des logiciels d'automatisme sur un système complet avec clavier, écran, imprimante et cassette, à l'aide de programmes puissants tels qu'assembleurs, désassembleurs et compilateurs, puis à transférer le résultat de ce travail sur une carte ne comportant que le matériel strictement nécessaire, mais permettant un certain confort d'utilisation.

Bien qu'il ne soit guère dans nos habitudes de parler prix, nous pensons intéressant d'effectuer une comparaison entre les deux solutions envisageables. Bien sûr, en ces temps économiquement incertains, des variations pourront être enregistrées, mais au moins le rapport restera-t-il valable :

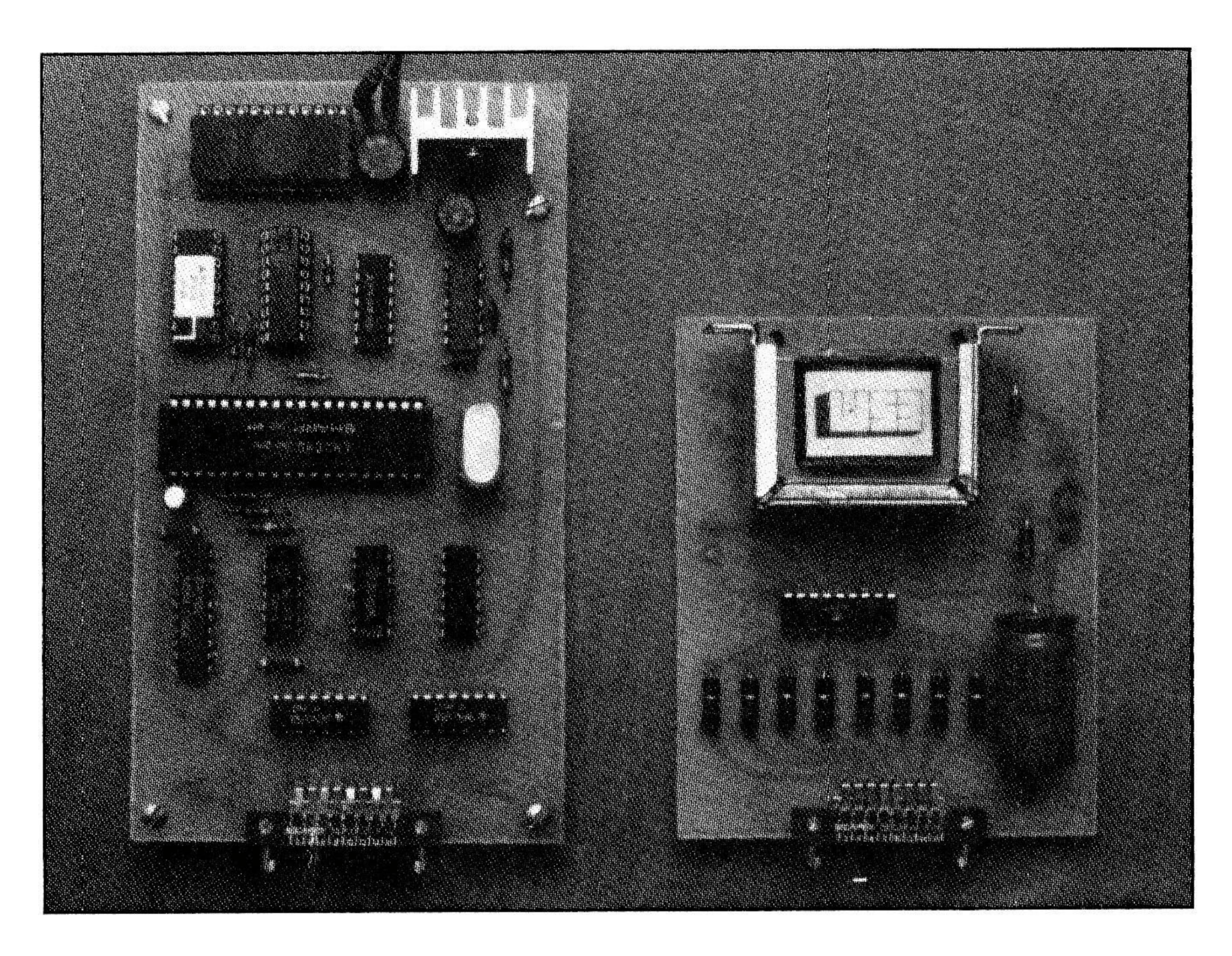
a) considérons le *système minimum* composé d'un ZX 81 et de la moins chère des cartes d'entrée-sortie : il faut compter au moins 880,00 F, sans parler du téléviseur et du magnétophone ;

b) choisissons la version la plus complète de la carte microprocesseur qui va être décrite (1 K RAM, 2 K ROM, 8 entrées, 8 sorties), nous arrivons tout juste à 300,00 F, somme qui se réduit à 250,00 F si l'on peut se passer de RAM ce qui, nous le verrons, est fréquent.

Qui plus est, une carte bien conçue démarrera l'exécution du programme dès sa mise sous tension, sans aucune intervention. Enfin, l'encombrement de ce « système maximum » se limite à celui d'un circuit imprimé de 150 × 85 mm, éventuellement complété par une alimentation 6 V, 9 V ou 12 V, qu'importe !

Conception générale de la carte

Il s'agit de rassembler, sur une carte homogène, un microprocesseur Z 80, des mémoires ROM et RAM, quelques circuits logiques et des interfaces d'entrée-sortie. C'est l'architecture quasi-invariable de tout



Notre carte microprocesseur et son module alimentation/étages de puissance.

système micro-informatique, mais dans le cas qui nous intéresse, il faut absolument garantir la compatibilité avec le ZX 81, de façon à assurer la « portabilité » des logiciels qui seront mis au point sur cette machine.

Il ne saurait bien sûr être question de copier le schéma du ZX qui utilise, de toute façon, un circuit intégré spécialement fabriqué pour Sinclair et donc introuvable, mais il est indispensable de respecter le même plan d'occupation mémoire. Comme Sinclair, nous réserverons les 16 premiers K-octets à la ROM, même si 2 K nous suffisent amplement (un boîtier d'EPROM 2716). Egalement, nous devons affecter à nos entrées-sorties des ports compatibles avec les cartes d'interface destinées au ZX 81. Nous avons donc choisi le port 127, ce qui correspond aux premières versions des cartes 8ES (restons français, que diable !).

Chacun aura compris que la différence fondamentale entre le ZX 81 et notre carte réside dans le fait que celle-ci ne dispose pas du moniteur BASIC Sinclair. Dans sa ROM sera donc directement logé le programme

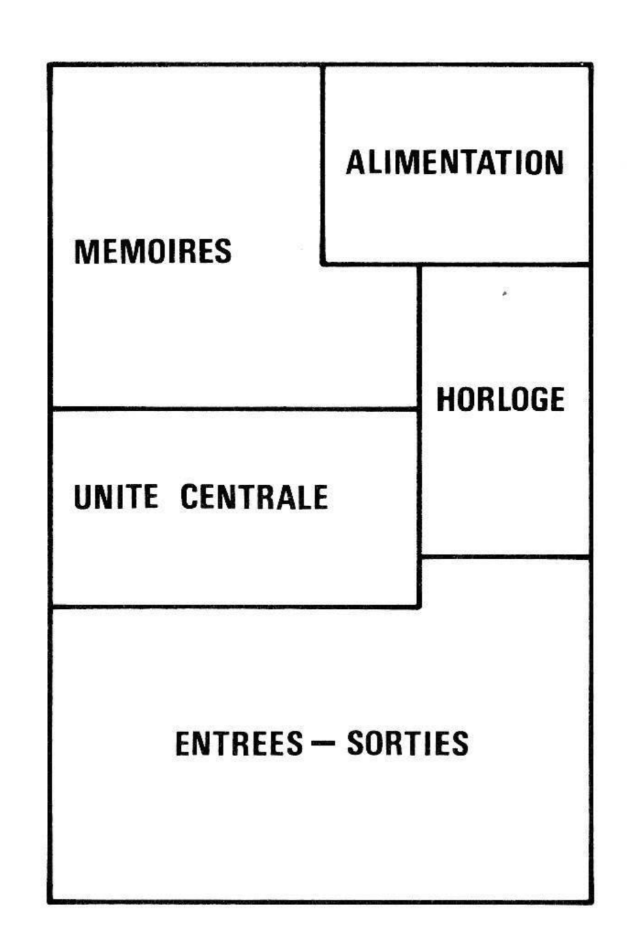


Fig. 9-1. *Organisation de la carte.*

« utilisateur », écrit en langage machine, ou à la rigueur en une forme convenable de BASIC *compilé*. Un microprocesseur Z 80 commençant toujours, lors de sa mise sous tension, par exécuter le programme débutant à l'adresse Ø, il est clair que le système bâti autour de la carte pourra démarrer absolument seul.

Que les nostalgiques du BASIC se consolent, il leur reste la possibilité, même si ce n'est pas très glorieux, de recopier dans leur EPROM certaines routines de la ROM Sinclair...

La *figure 9-1* résume cette organisation générale, en conformité avec l'encombrement qui a été défini plus haut. La *figure 9-2*, pour sa part, donne le brochage des principaux circuits intégrés utilisés afin d'éclairer le schéma complet de la *figure 9-3*.

Le schéma de principe

L'essentiel des liaisons entre les boîtiers consiste à distribuer les bus d'adresses et de données entre les divers circuits, ce qui n'appelle pas de commentaire. On peut, par contre, s'intéresser plus particulièrement aux circuits dits annexes qui, finalement, déterminent les caractéristiques du système.

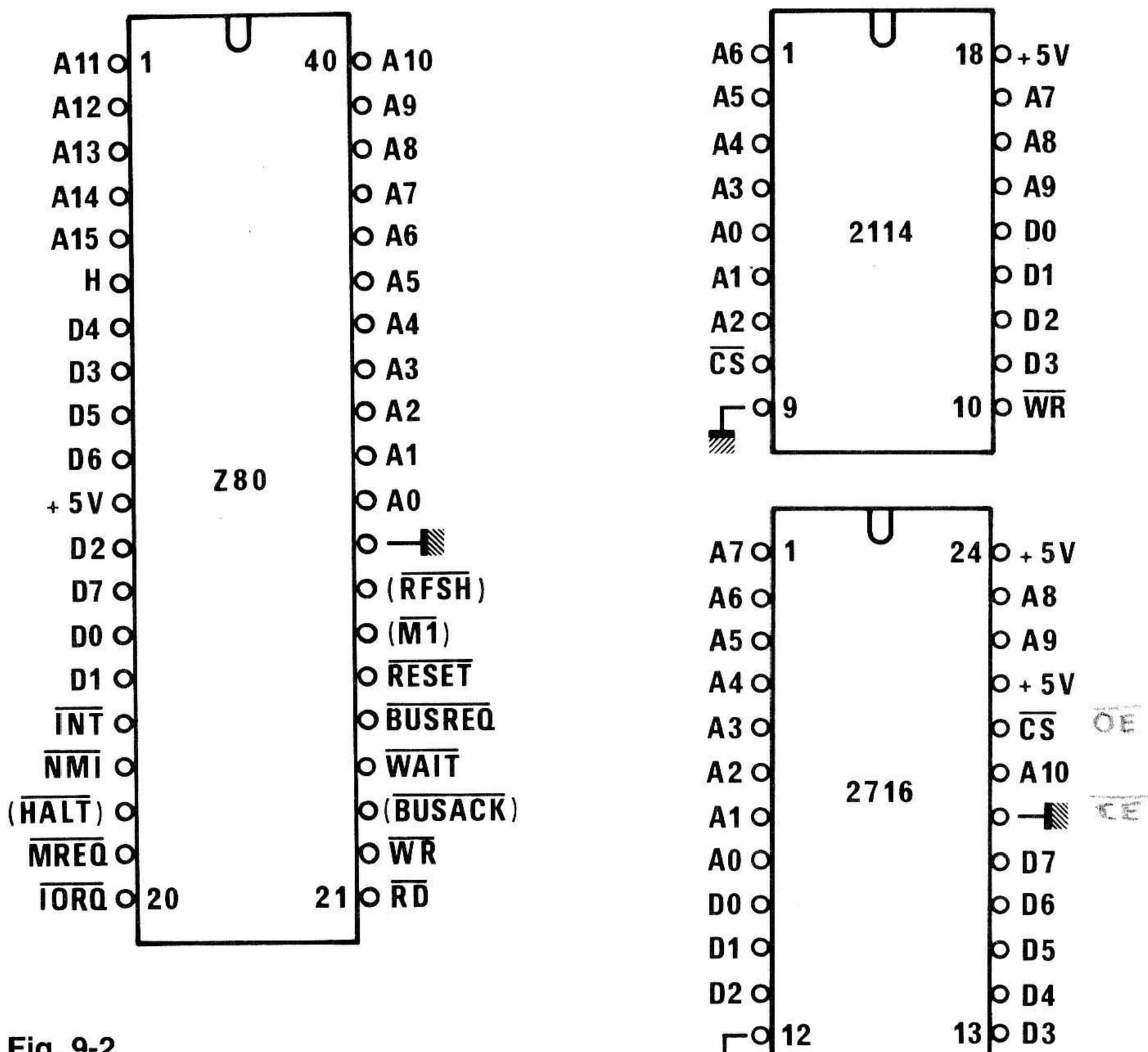


Fig. 9-2.
Brochage des circuits intégrés spécifiques.

• L'alimentation

Tous les circuits de la carte fonctionnent sous un + 5 V unique, et consomment au maximum 300 mA. Comme il est important, pour la santé du matériel, que cette tension soit très bien régulée, nous avons prévu, sur la carte même, un régulateur intégré de type 7805, équipé d'un refroidisseur conséquent permettant à la tension d'entrée de monter si nécessaire bien au-delà de 12 V (utilisation directe sur batterie auto, par exemple). Des condensateurs de 10 nF sont répartis en découplage aux points névralgiques des lignes de distribution.

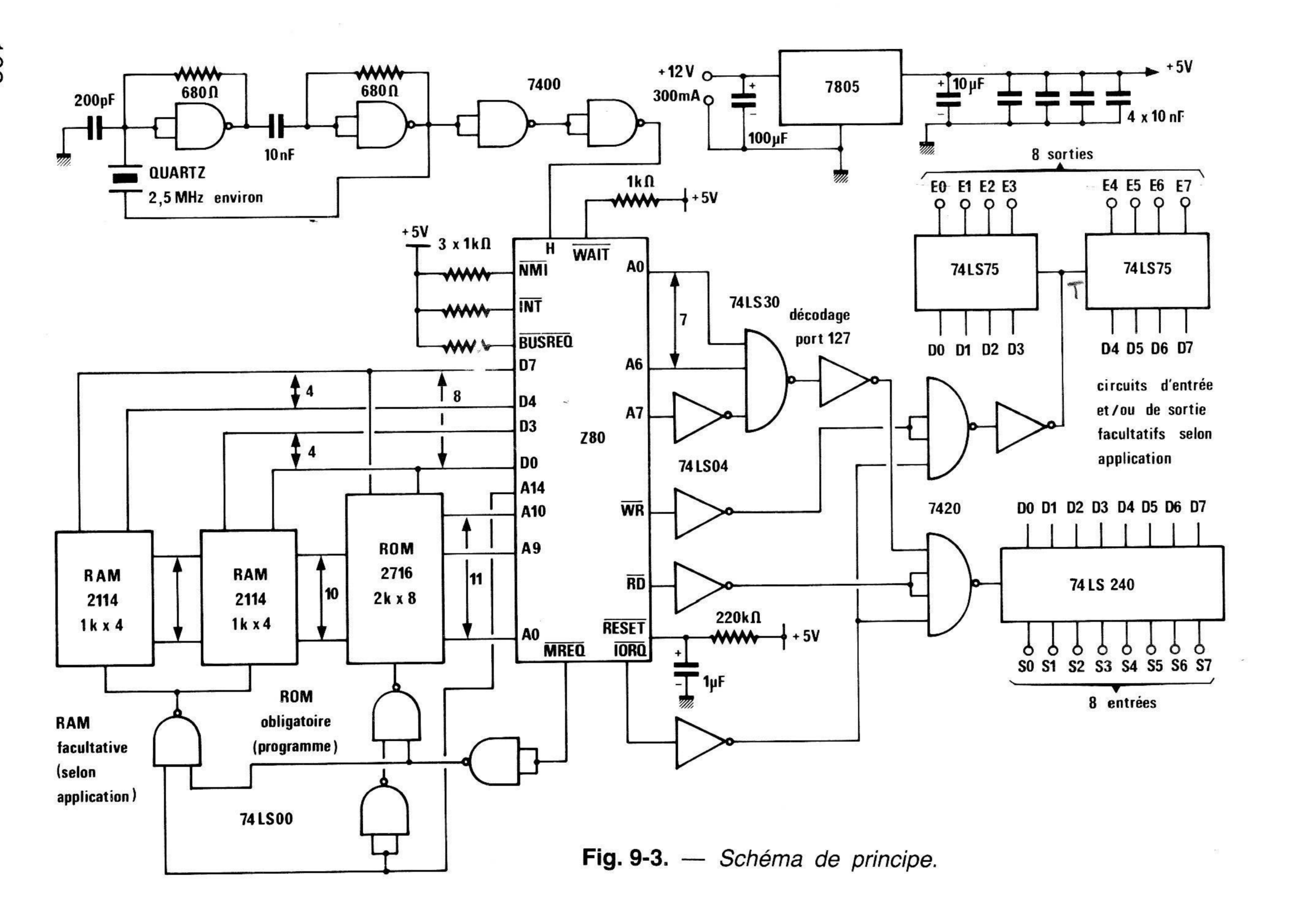


Fig. 9-3. — *Schéma de principe.*

- **L'horloge**

Tout microprocesseur doit être piloté par une horloge délivrant un signal de haute qualité (fronts raides, rapport cyclique précis) et de fréquence très stable si l'on souhaite pouvoir compter sur la précision des temporisations qu'il aura à exécuter.

Nous avons donc utilisé un classique oscillateur à quartz, dont le signal est mis en forme par deux portes inverseuses en cascade.

La fréquence choisie est d'environ 2,5 MHz (contre 3,25 dans le ZX 81) afin de permettre l'emploi d'un Z 80 ordinaire, plus économique encore que largement suffisant. Il faudra tenir compte de cette différence lors de l'écriture d'éventuels programmes comportant des temporisations.

Il reste bien sûr possible de monter un quartz de 3,25 MHz et un Z 80 A, mais le jeu n'en vaut pas forcément la chandelle.

- **Les sélections mémoire**

Un simple boîtier 74LS00 élabore des signaux analogues à ceux nommés $\overline{\text{RAMCS}}$ et $\overline{\text{ROMCS}}$ sur le ZX 81, et qui sélectionnent la ROM pour A14 = Ø, et la RAM pour A14 = 1, étant entendu que $\overline{\text{MREQ}}$ doit aussi être à Ø (demande d'accès à la mémoire).

- **Les entrées-sorties**

Nous voici au cœur de toute application d'automatisme, pour laquelle les accès à l'extérieur sont indispensables. Un décodage des huit lignes basses du bus d'adresses sert à élaborer le signal correspondant à l'octet 127 (numéro de port). On notera que, même si cela utilise beaucoup de matériel (un 74LS30 entier plus un inverseur), il a été décidé de procéder à un décodage complet, évitant toute ambiguïté avec d'autres numéros de ports.

Ce signal de validation est ensuite combiné avec $\overline{\text{IORQ}}$ (demande d'accès à un port), puis avec $\overline{\text{WR}}$ et $\overline{\text{RD}}$, de façon à obtenir respectivement les impulsions de validation des dispositifs de sortie et d'entrée (en effet, l'unité centrale *écrit* sur une sortie, mais *lit* une entrée).

Les circuits de sortie sont bâtis autour de deux quadruples latches 74LS75, capables de mémoriser aussi longtemps que nécessaire les états des huit sorties très brièvement transmis par le bus de données.

Les signaux d'entrée, pour leur part, transitent par huit buffers à trois états contenus dans un seul circuit 74LS240 ou 74LS244, selon que l'on désire que l'entrée soit complémentée ou non. Notons que le même choix

existe au niveau des sorties, puisque les 74LS75 disposent chacun de quatre sorties directes et de quatre sorties complémentées.

Ces possibilités de choix seront précieuses lors du raccordement des dispositifs utilisateurs (circuits de puissance), dont la diversité interdit l'intégration sur la carte. Les huit entrées et les huit sorties sont donc directement aux niveaux TTL LS.

Réalisation pratique

Industriellement, la réalisation d'une carte comportant un nombre aussi considérable de liaisons ferait appel à la technique du circuit imprimé double face à trous métallisés (comme le ZX). Ce procédé étant virtuellement inaccessible à nos lecteurs, il fallait soit choisir la technique double face classique, au prix d'un nombre prohibitif de traversées et même de straps, soit se tourner vers... autre chose !

Autre chose, c'est le circuit simple face complété par des bus en fil émaillé soudable côté cuivre. Cette technique proche du wrapping, mais se contentant d'un outillage très bon marché, pourrait sembler relever du bricolage le plus infâme si l'aéronautique ne l'employait massivement pour ses réalisations à haute fiabilité !

Le plus commode est d'utiliser un « stylo à câbler » muni d'une bobine de fil spécial dont la double couche d'émail fond au seul contact de l'étain liquide (SIEMENS — SEDI). A défaut, on pourra se contenter de fil émaillé courant, mais le dénudage manuel sera plus fastidieux, tout en entraînant des risques de blessure du brin de cuivre.

Une fois ce câblage achevé, on pulvérisera un vernis à séchage rapide tel que le TROPICOAT JELT, de façon à assurer la cohésion parfaite de l'ensemble.

Avec cette technique, il est absolument indispensable d'utiliser des supports pour les principaux circuits intégrés (microprocesseur et mémoires), et si possible pour les boîtiers d'entrée-sortie les plus exposés à des incidents sérieux. A défaut, leur éventuel remplacement poserait de très délicats problèmes. Pour l'EPROM, destinée à contenir le logiciel, et donc à être fréquemment échangée, on choisira un modèle de très bonne qualité, à force d'insertion très réduite. Selon l'application prévue, on pourra se dispenser de câbler les circuits éventuellement superflus : beaucoup de logiciels fonctionnent sans RAM (sur les seuls registres internes du Z 80), alors que d'autres n'utilisent que des circuits de sortie, et pas toujours en grand nombre (souvent pas plus de quatre).

Dans de tels cas, on pourra choisir tout simplement de laisser vides les supports concernés (afin de conserver la possibilité d'une future

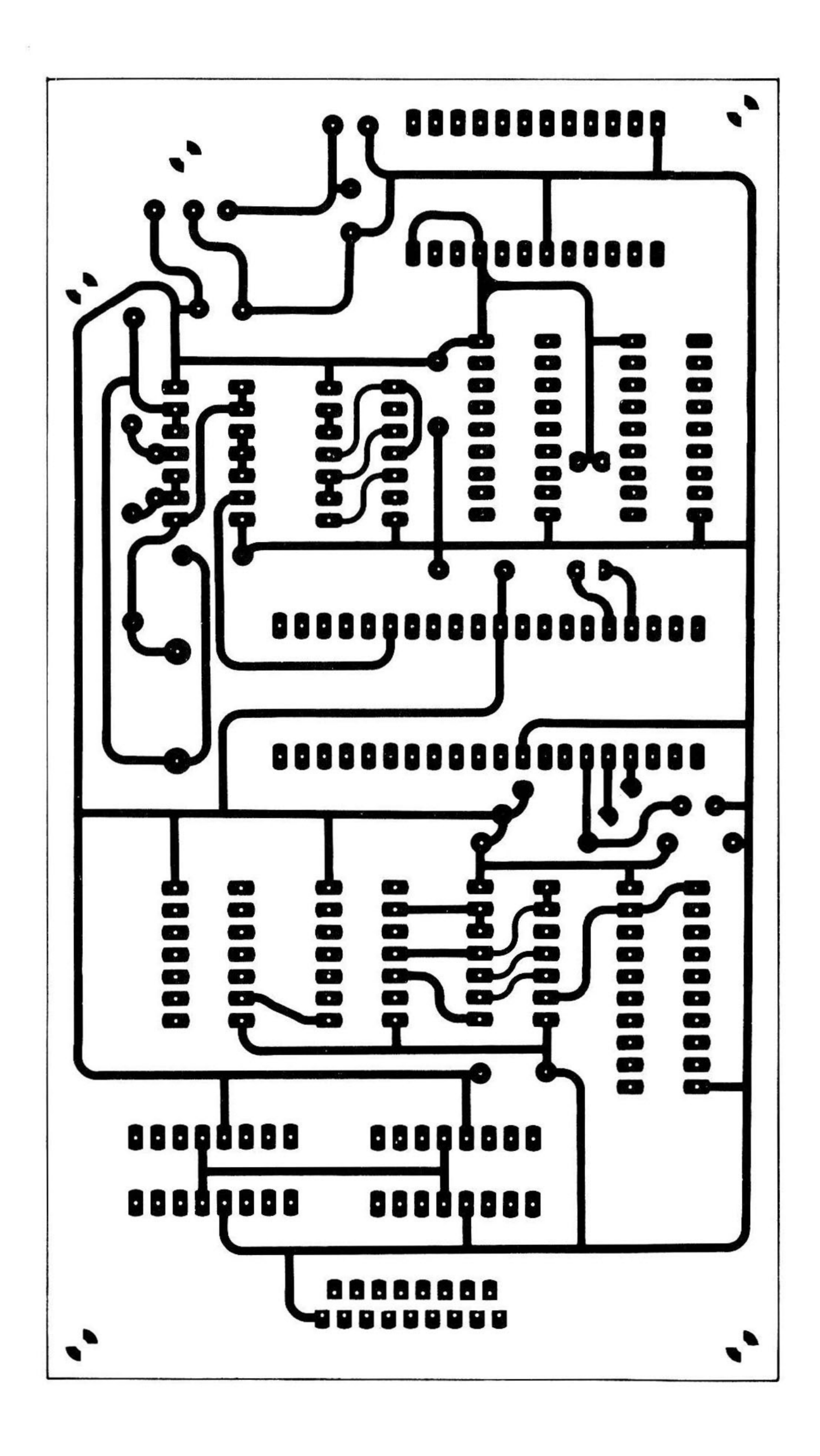


Fig. 9-4. — *Circuit imprimé.*

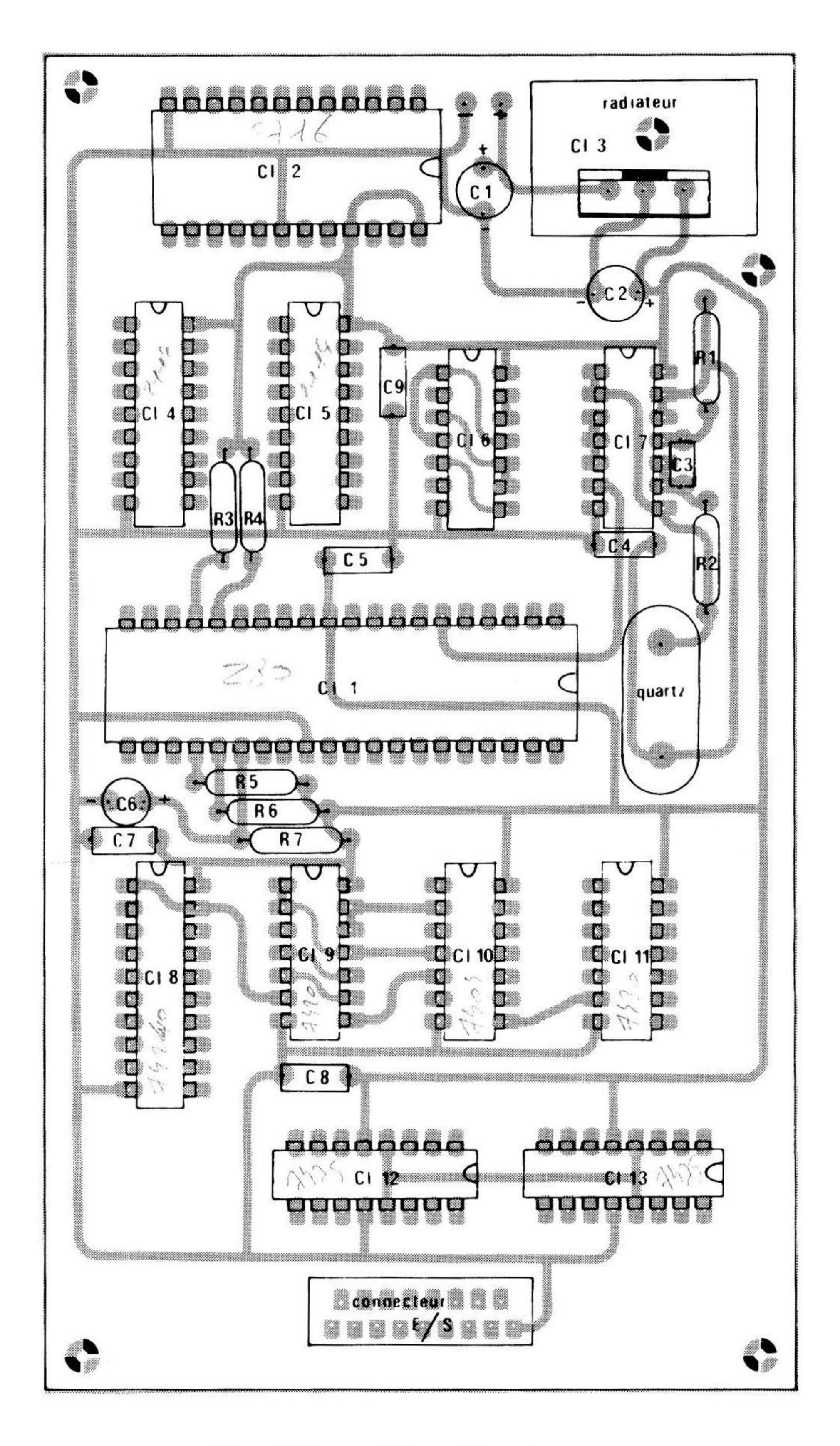


Fig. 9-5. — *Plan d'implantation*
(voir nomenclature en page 175).

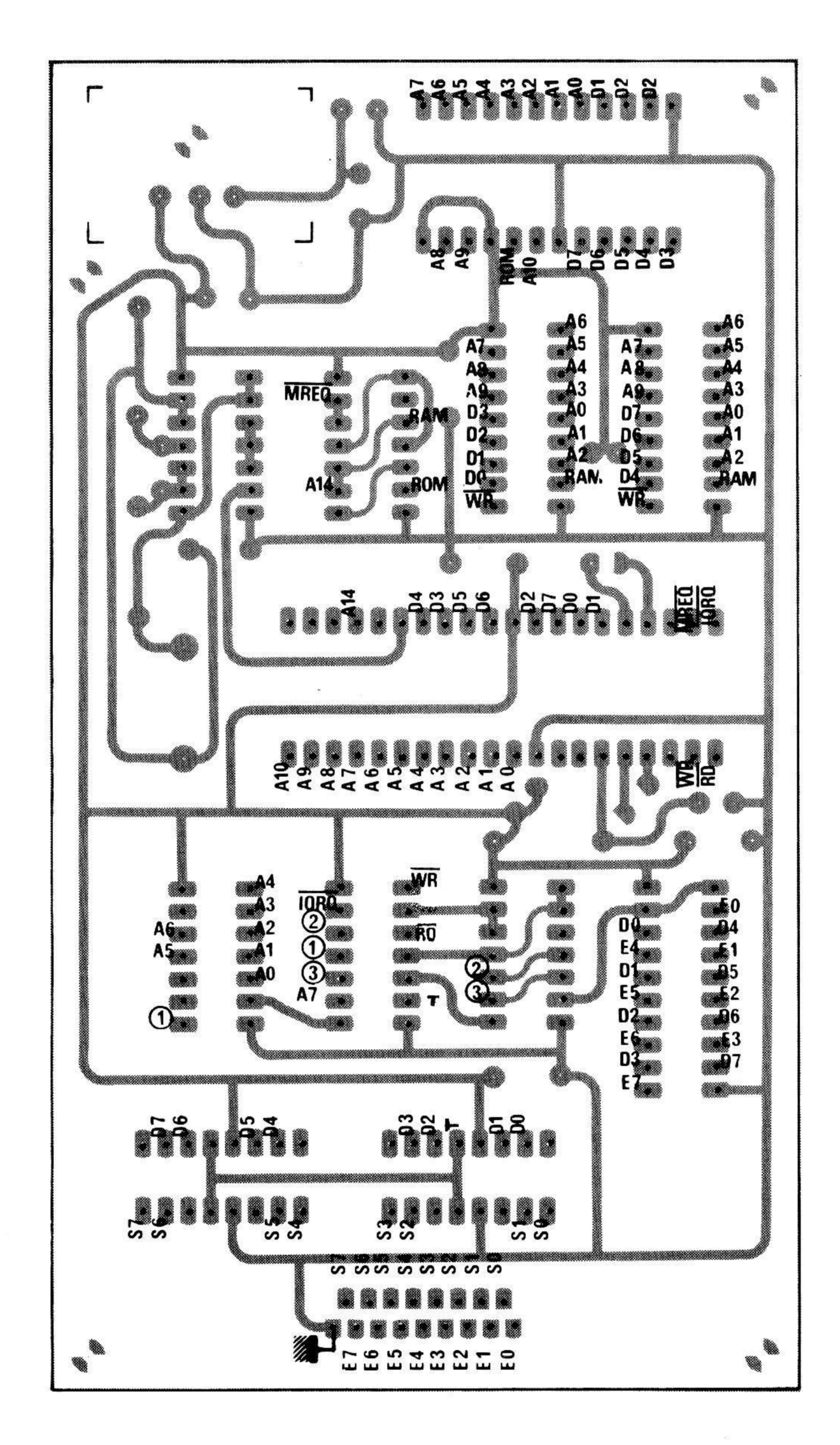


Fig. 9-6. — *Les liaisons filaires à effectuer (relier les points de même nom). Attention : les broches repérées par un point seul ne reçoivent aucune liaison.*

extension de configuration), ou bien on omettra carrément tout le câblage correspondant, à des fins de simplification et d'économie.

Il résulte de cette conception particulière que les plans de câblage sont au nombre de trois :

La *figure 9-4* donne le tracé du circuit imprimé simple face dont la réalisation par des moyens « amateur » ne pose pas le plus petit problème. On pourra éventuellement modifier l'implantation du connecteur d'entrée-sortie, selon les besoins particuliers de chacun. Notre maquette utilise un modèle de fabrication SOCAPEX, muni de 17 contacts (8 entrées, 8 sorties, et la masse). L'alimentation est amenée séparément au moyen de deux cosses poignard mais pourrait, si nécessaire, être ramenée sur un même connecteur.

Le plan d'implantation de la *figure 9-5* ne soulève pas de remarque particulière, si ce n'est, bien sûr, que les circuits MOS ne doivent être montés dans leurs supports qu'au terme de *toutes* les opérations de câblage.

C'est avec la *figure 9-6* que nous abordons véritablement les choses sérieuses, puisque ce document indique la totalité des liaisons filaires à effectuer. Les risques d'erreur sont très minces, puisque l'essentiel de ce câblage concerne les bus. Toute erreur déboucherait rapidement sur l'impossibilité de réaliser l'une des liaisons suivantes, d'où un autocontrôle permanent des opérations.

Le travail consiste à réunir par un même fil, tous les points portant le même repère numérique, alphabétique, ou alphanumérique. Bien souvent, ces points sont en nombre supérieur à deux, et il est alors commode de ne pas couper le fil, cette démarche « de porte à porte » faisant gagner du temps et diminuant les risques de soudures défectueuses.

Une bonne précaution consiste à contrôler le câblage à l'ohmmètre *avant toute insertion de circuits intégrés*, mais il est encore plus important de se livrer à un test visuel de la qualité des soudures (notamment absence de court-circuits). A part la présence du + 5 V et du signal d'horloge, on ne peut guère tester la carte qu'en lui faisant exécuter un programme, qu'il faut au préalable charger dans une mémoire EPROM de type 2716.

Programmation de la carte

Il existe principalement deux moyens permettant d'obtenir un logiciel pour cette carte : utiliser l'un des programmes publiés pour elle (et pas pour une autre, sauf modifications), ou bien mettre au point soi-même un tel logiciel sur un ZX 81 considéré alors comme un système de

développement, et donc muni des accessoires voulus (carte 8ES et logiciels de programmation en assembleur).

Seulement, dans ce dernier cas, il n'est pas possible de tester sur le ZX 81 les programmes dans la zone mémoire qui leur sera dévolue sur la carte (Ø à 2047). En effet, cette zone correspond au début de la ROM Sinclair, qu'il n'est *ici* pas question de supprimer !

Fort heureusement, à condition de respecter certaines règles très simples, notre carte pourra exécuter sans coup férir des programmes mis au point dans certaines zones de la RAM du ZX 81, et en particulier dans l'espace compris entre les adresses 8192 et 10239. L'utilisation de cette zone exige le recours à certains artifices, tels que le blocage de la ROM pour des adresses tombant dans cette fourchette, et son remplacement par de la RAM (voir chapitre 4).

Bien mieux, si l'on prend soin de n'écrire que des programmes « relogeables » (en adressage relatif), la zone de mémoire dans laquelle ils auront été mis au point n'aura aucune incidence sur le fonctionnement de la carte.

De toute façon, on retiendra de ce qui précède qu'il ne faut pas s'étonner de trouver dans des logiciels écrits pour cette carte, des renvois à des adresses inexistantes en EPROM : le décodage spécial de la ligne A14 se charge de rétablir le bon « aiguillage ». Voici tout le secret de la compatibilité avec notre carte des logiciels écrits pour le ZX 81.

Le programme de test de la carte n'échappe pas à cette règle. Implanté à partir de l'adresse Ø de l'EPROM, il se compose en tout et pour tout de sept octets, dont voici les codes décimaux : 219, 127, 211, 127, 195, Ø, 32.

On reconnaît ici	IN A, (127)
les trois instructions suivantes	OUT (127), A
de l'assembleur Z 80 :	JP 8 192

La carte effectue donc une entrée sur le port n° 127, pour ressortir aussitôt le même octet (temporairement stocké dans l'accumulateur), sur le même port. Un bouclage s'effectue alors sur l'adresse 8192, mais la carte revient en fait à l'adresse 0, puisque la ligne A13, de poids 8192, n'est pas reconnue par l'EPROM.

Pour vérifier le bon fonctionnement de la carte, il suffit de mettre successivement à la masse les huit entrées, en vérifiant que les sorties suivent bien individuellement ces changements d'états.

Le programmateur décrit au chapitre 3 « figera » très rapidement ces quelques octets au début d'une 2716 qui, effacée à l'issue de cette vérification de routine, pourra être réutilisée pour accueillir un logiciel plus utilitaire.

Les différents programmes rédigés en langage machine que nous avons fournis au chapitre précédent peuvent directement « tourner » sur cette carte, ainsi que toute réalisation personnelle de nos lecteurs conforme aux règles énoncées. Reste donc à incorporer cette carte dans un ensemble complet et homogène comportant une alimentation et des circuits de puissance.

Un module alimentation et étages de puissance

La carte microprocesseur dont nous avons décrit la réalisation dans ce chapitre constitue le « tronc commun » de toutes les applications envisageables, puisque le simple chargement d'un logiciel approprié suffit à déterminer entièrement le comportement du système.

Autour de ce module de base peuvent venir se greffer divers circuits périphériques rendus nécessaires par telle ou telle application particulière.

Les deux principales extensions auxquelles il peut être utile de recourir sont l'alimentation secteur et les étages de sortie de puissance.

Un circuit d'alimentation secteur secouru

Notre carte microprocesseur fonctionne sous 5 volts et absorbe environ 300 mA. Cependant, un régulateur de tension incorporé permet l'usage de toute tension continue comprise entre 6 et 15 volts environ.

Bien des sources de tension répondent à ces caractéristiques, mais c'est bien souvent au secteur 220 V que l'on souhaite confier l'alimentation d'un montage pratique.

L'adaptation est extrêmement simple, puisqu'il n'y a pas lieu de procéder à une stabilisation de tension : un redressement double alternance et un filtrage sommaire suffisent, comme en témoigne le schéma de la *figure 9-7.*

Une petite particularité, cependant, très utile lorsque le microprocesseur exécute des tâches de grande importance : une diode supplémentaire est prévue, qui permet de connecter une alimentation de secours dont la tension sera comprise entre 6 et 9 volts. Ainsi, en cas de défaillance du secteur, la commutation instantanée par les diodes évitera tout incident de fonctionnement.

Lorsque l'alimentation principale est présente, on mesure au moins 10 volts aux bornes du condensateur de 2200 μF, et la diode de secours

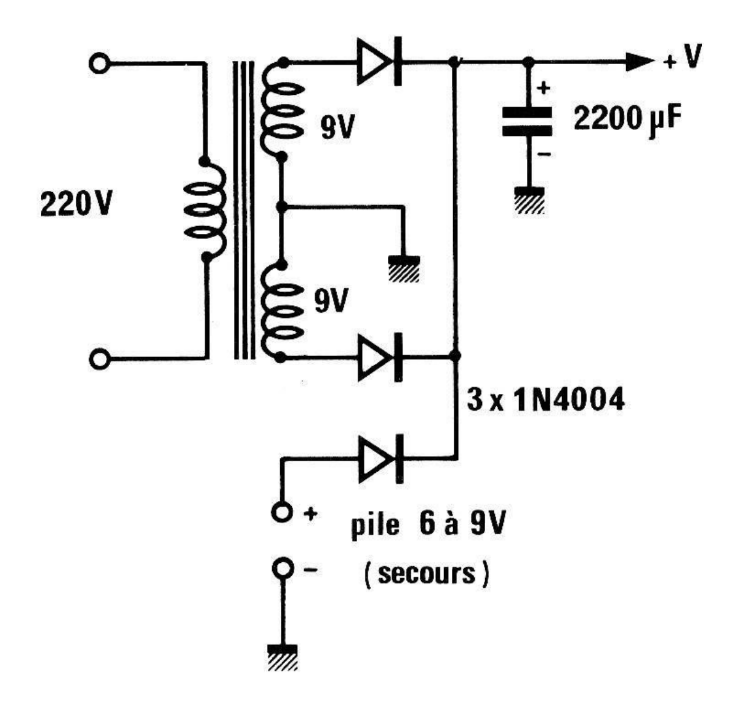


Fig. 9-7. — *Schéma de l'alimentation secteur.*

est donc absolument bloquée, isolant la pile ou la batterie de secours. Il est important de ne pas dépasser 9 volts, car alors cette déconnexion ne serait plus garantie. Une solution luxueuse consiste à employer une batterie au cadmium-nickel montée en tampon aux bornes d'un chargeur.

Le plus souvent, on peut cependant se contenter d'une de ces très grosses piles 6 volts qui équipent les balises de chantier ou les lampes de camping : de nombreuses heures d'autonomie sont ainsi garanties.

Des étages de sortie à hautes performances

Les circuits TTL dont sont équipées les huit sorties de la carte microprocesseur peuvent commander directement un certain nombre de dispositifs externes, tels que des diodes LED ou certains relais REED.

Lorsque des puissances plus notables sont en jeu, il faut passer par des étages de puissance capables de commuter un courant non négligeable sous une tension nettement supérieure à 5 volts.

On pourrait songer à réaliser huit circuits identiques au moyen de composants discrets, mais les transistors, les résistances et les diodes de protection finiraient par tenir beaucoup de place, et par coûter assez cher.

Il est beaucoup plus élégant de faire appel à l'un de ces circuits intégrés spécialement étudiés pour suivre les microprocesseurs, et incorporant huit étages Darlington protégés dans un boîtier à 18 broches.

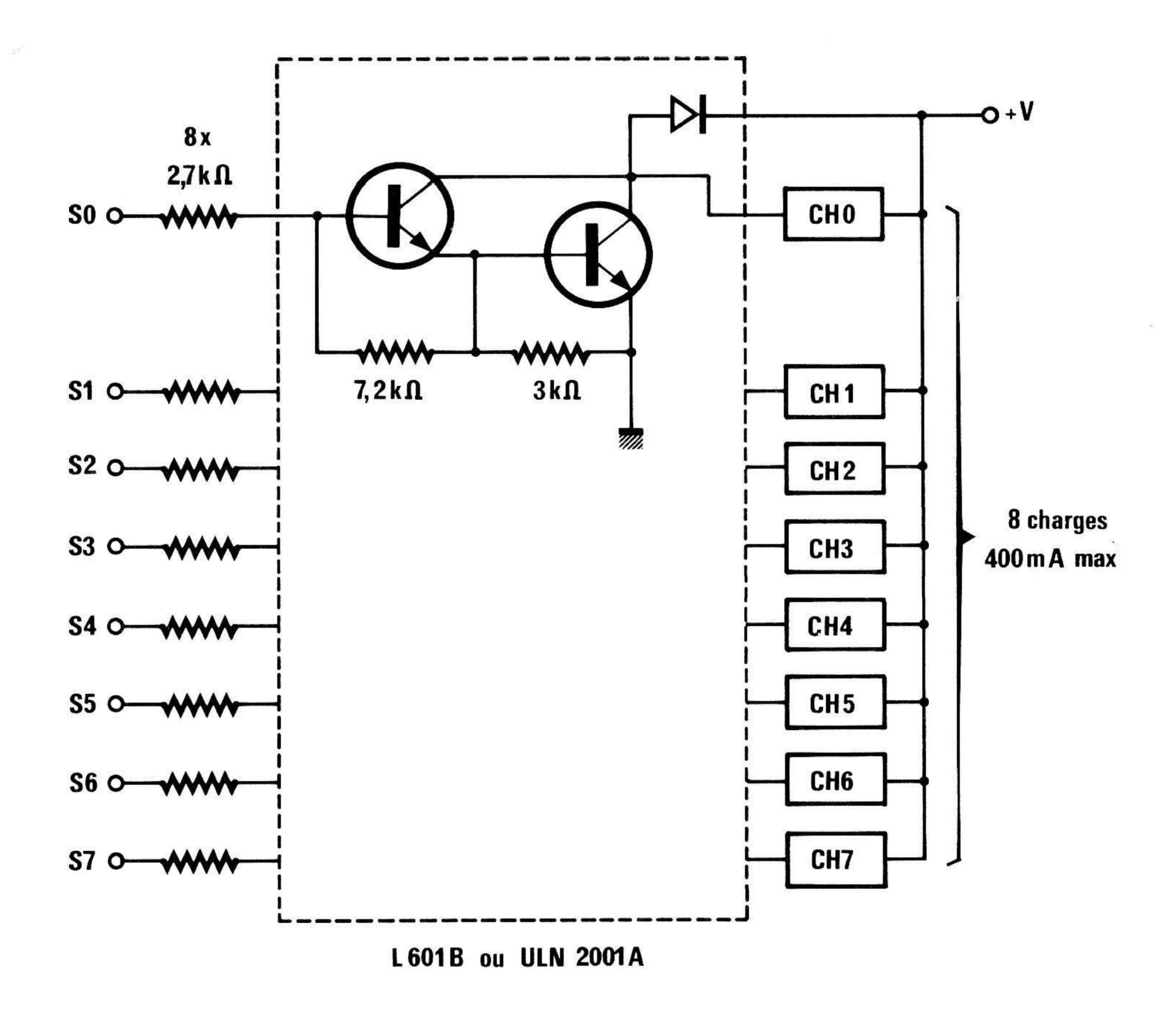


Fig. 9-8. — *Schéma des étages de sortie de puissance.*

Notre choix s'est porté sur le type le plus répandu, existant d'ailleurs chez plusieurs fabricants. On pourra ainsi utiliser tout à fait indifférement le L 601 B de SGS-Atès ou le ULN 2001 A de SPRAGUE.

A partir de niveaux TTL transmis par des résistances de protection, ces circuits peuvent commander huit charges quelconques, même selfiques, consommant au maximum 400 mA (600 en pointe) sous une tension pouvant atteindre 90 volts. Le montage de la *figure 9-8* étant jumelé avec l'alimentation de la *figure 9-7*, il est bien évident qu'il ne pourra être question de dépasser les possibilités du transformateur utilisé en commandant, par exemple, huit ampoules de 400 mA. Toutefois, en

cas de nécessité, il sera extrêmement simple de séparer les étages de sortie de l'alimentation pour y substituer une source d'énergie extérieure. On veillera alors à bien relier la broche 10 du circuit intégré à l'alimentation des charges, et pas à une autre !

Réalisation pratique

Le circuit imprimé de la *figure 9-9* est prévu pour recevoir tous les composants du montage, y compris un transformateur ESM de 2 fois 9 volts, et de puissance 5 VA.

Cette puissance suffit pour subvenir aux besoins de la carte et de quelques relais 12 volts.

Si des charges plus exigeantes devaient être commandées, on pourrait sans précaution particulière adapter un transformateur de puissance plus conséquente.

L'implantation d'un connecteur identique à celui de la carte microprocesseur (SOCAPEX 127-17 AF 1 YC) a été prévue en bord de carte, avec exactement le même brochage.

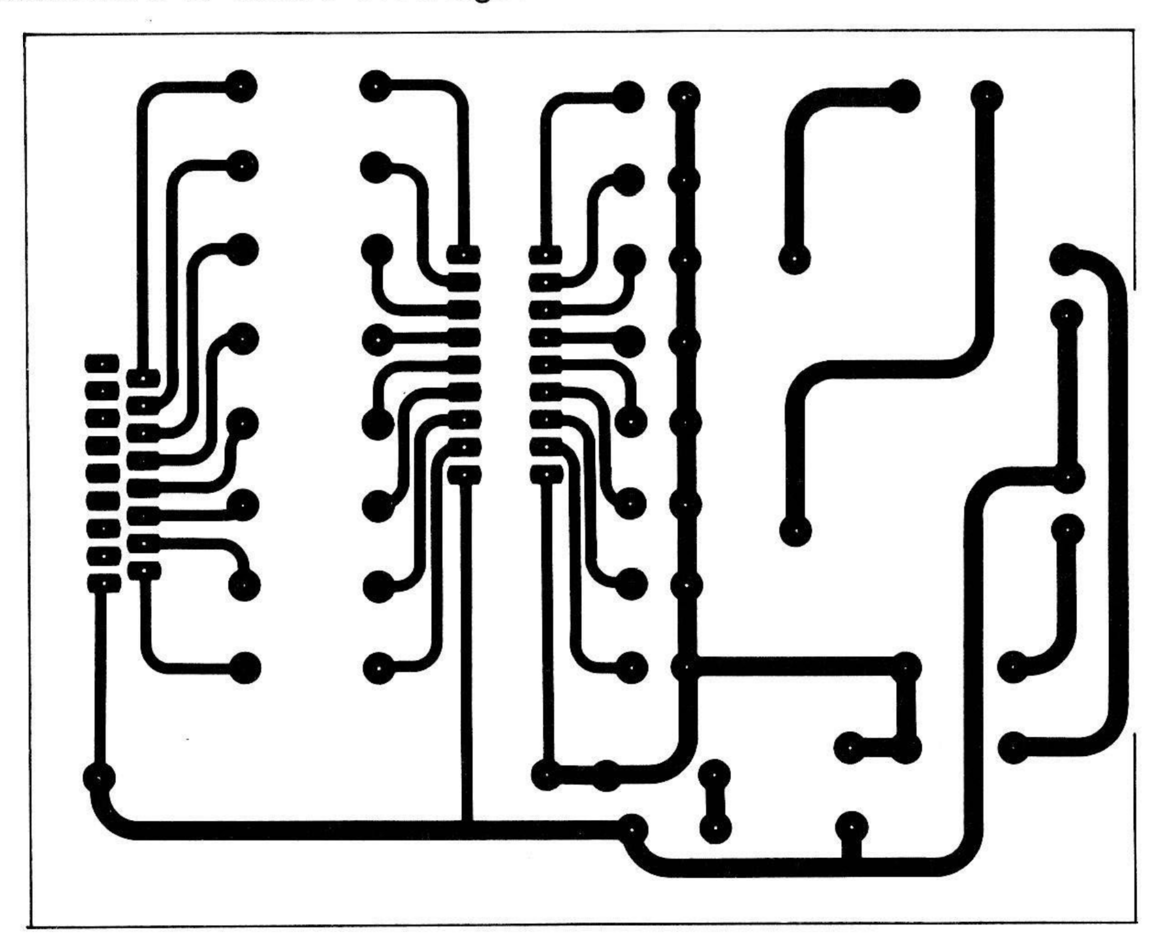


Fig. 9-9.

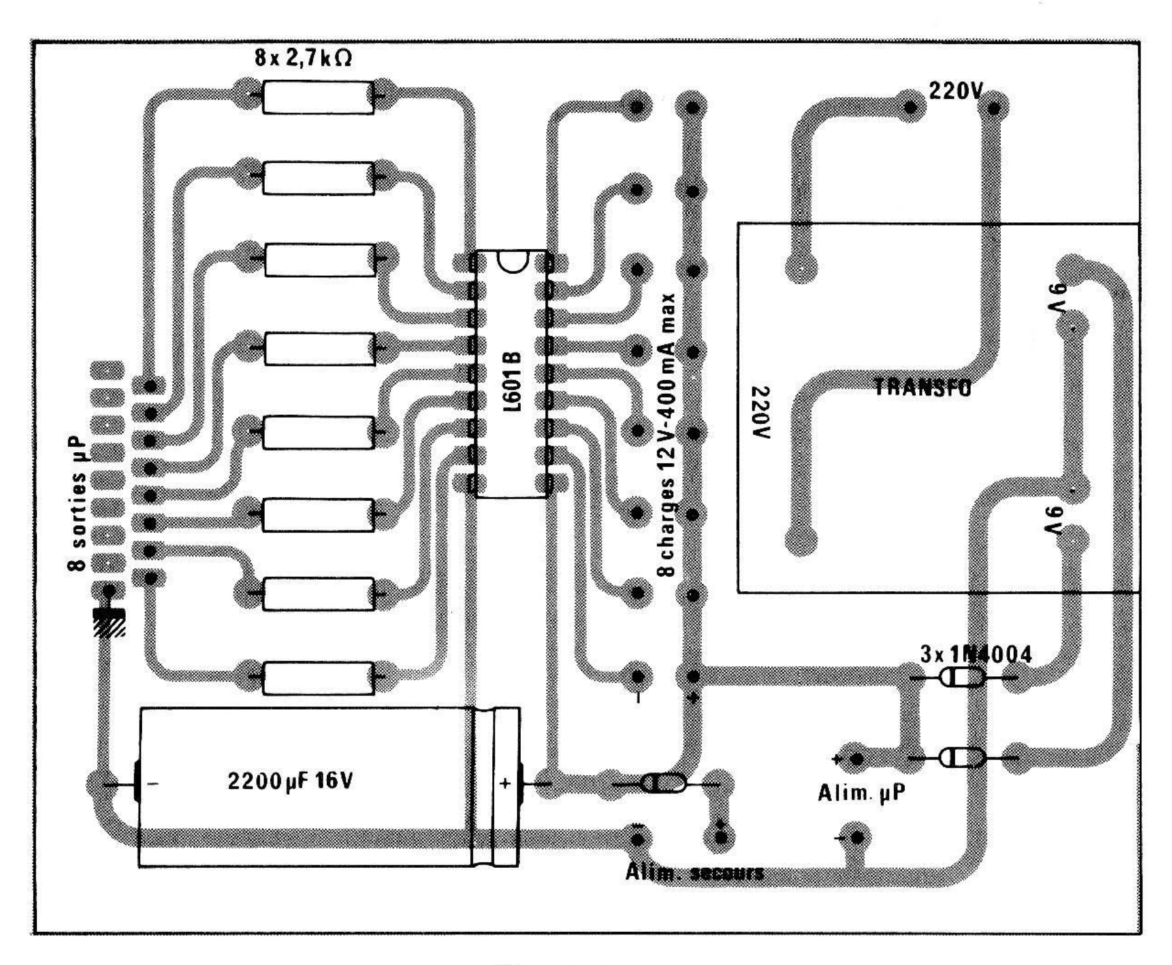


Fig. 9-10.

Cela ajouté au fait que la largeur des deux cartes est la même facilitera la réalisation d'un ensemble homogène dans un boîtage approprié muni d'un « fond de panier ».

Le plan de câblage de la *figure 9-10* ne nécessite pas de commentaire particulier, en raison de la simplicité des opérations d'assemblage.

Conclusion

Cette carte n'est bien évidemment pas indispensable pour faire fonctionner notre système à microprocesseur, auquel cas ses circuits auraient été incorporés au circuit de base. Elle facilite cependant grandement la commande, dans les meilleures conditions de sécurité et de fiabilité, de toutes sortes de charges appartenant à la famille des « actionneurs » : relais, contacteurs, petits moteurs, voyants, électro-aimants. Le fait que huit circuits de sortie soient offerts par un seul circuit intégré peu coûteux évite la tentation d'un câblage partiel et permet donc à l'utilisateur de disposer à tout moment de toutes les possibilités du microprocesseur pour n'importe quelle application, simple ou complexe.

Nomenclature de la carte microprocesseur

Résistances

R_1 : 680 Ω
R_2 : 680 Ω
R_3 : 1 kΩ
R_4 : 1 kΩ
R_5 : 1 kΩ
R_6 : 1 kΩ
R_7 : 220 kΩ

Condensateurs

C_1 : 100 μF 63 V
C_2 : 10 μF 63 V
C_3 : 10 nF
C_4 : 220 pF
C_5 : 10 nF
C_6 : 1 μF 63 V
C_7 : 10 nF
C_8 : 10 nF

Circuits intégrés

CI_1 : Z80 CPU
CI_2 : MM 2716
CI_3 : 7805
CI_4 : MM 2114
CI_5 : MM 2114
CI_6 : 74 LS 00
CI_7 : 74 LS 00
CI_8 : 74 LS 240 ou 74 LS 244 (voir texte)
CI_9 : 74 LS 20
CI_{10} : 74 LS 04
CI_{11} : 74 LS 30
CI_{12} : 74 LS 75
CI_{13} : 74 LS 75

Divers

1 quartz 2,5 MHz environ
1 connecteur 17 broches
1 refroidisseur pour 7805
Supports de CI

Achevé d'imprimer
sur les presses
de l'Imprimerie Marcel Bon
70001 Vesoul
Dépôt légal juin 1984
N° d'éditeur : 424
N° d'imprimeur : 2787

ROBOTISEZ VOTRE ZX 81

Ne vous débarrassez pas de votre ZX 81 ! Même s'il est un peu défraîchi, il conserve intacte sa puissance de traitement de l'information.

Vous pouvez **le transformer à l'aide de quelques accessoires** faciles à construire, **en un véritable « robot domestique ».** Sans écran TV ni magnétophone, il exécutera fidèlement une tâche programmée une fois pour toutes dans une mémoire permanente.

Principaux chapitres :

- Autopsie du ZX 81
- Une nouvelle vie
- Des entrées et des sorties
- Réorganisons la mémoire
- Programmons nos mémoires mortes
- Un afficheur autonome
- Une carte sonore
- Des applications pratiques
- Une carte microprocesseur

ISBN 2-85535-076-X

TARIF CODE Q


12
ROBOTISEZ VOTRE ZX 81
P. GUEULLE
ETSF